POP methodology

Marta Garcia-Gasulla, BSC

EU H2020 Centre of Excellence (CoE)

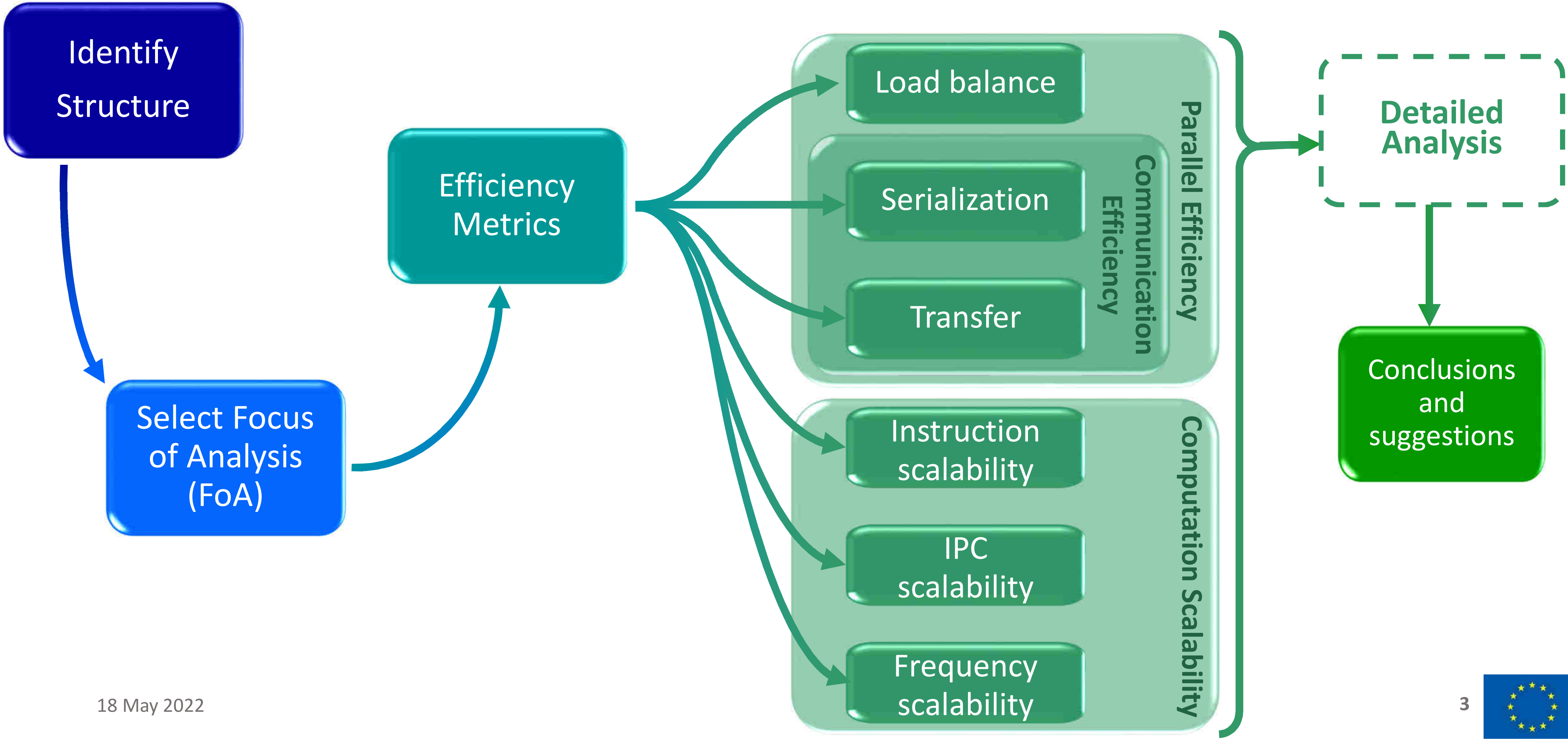
1 December 2018 – 30 November 2021

Grant Agreement No 824080

Content

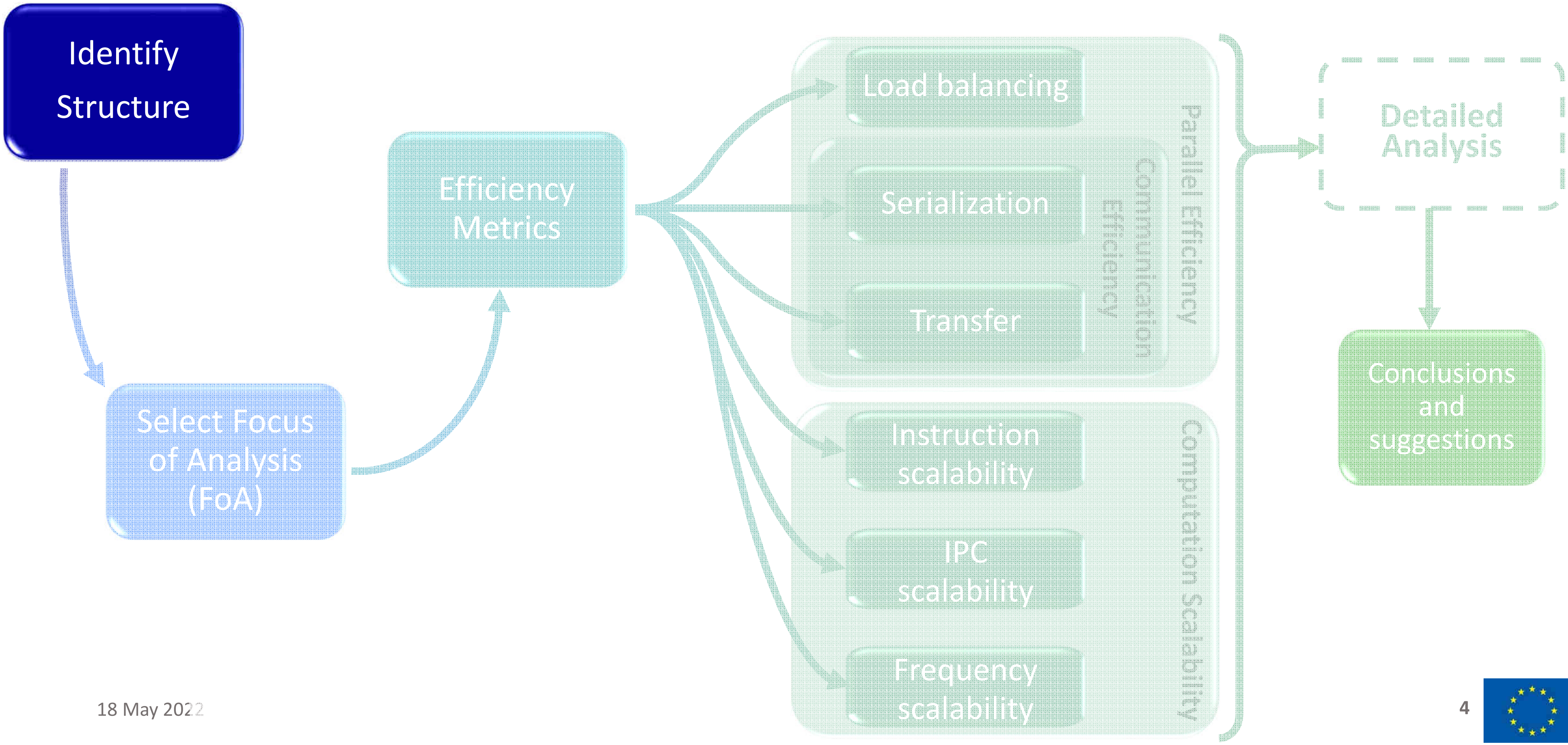
- POP methodology
  - A methodology for performance analysis
  - Agnostic of the tools
    - ➢ But this presentation is based on BSC tools (Extrae, Paraver, Dimemas and ModelFactors)
- In this presentation MPI only applications
  - But POP methodology is being extended to:
    - ➢ Hybrid models: OpenMP, GPUs…
    - ➢ I/O
    - ➢ Vectorization
  - These can be added into an advanced version of this course

The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balance
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

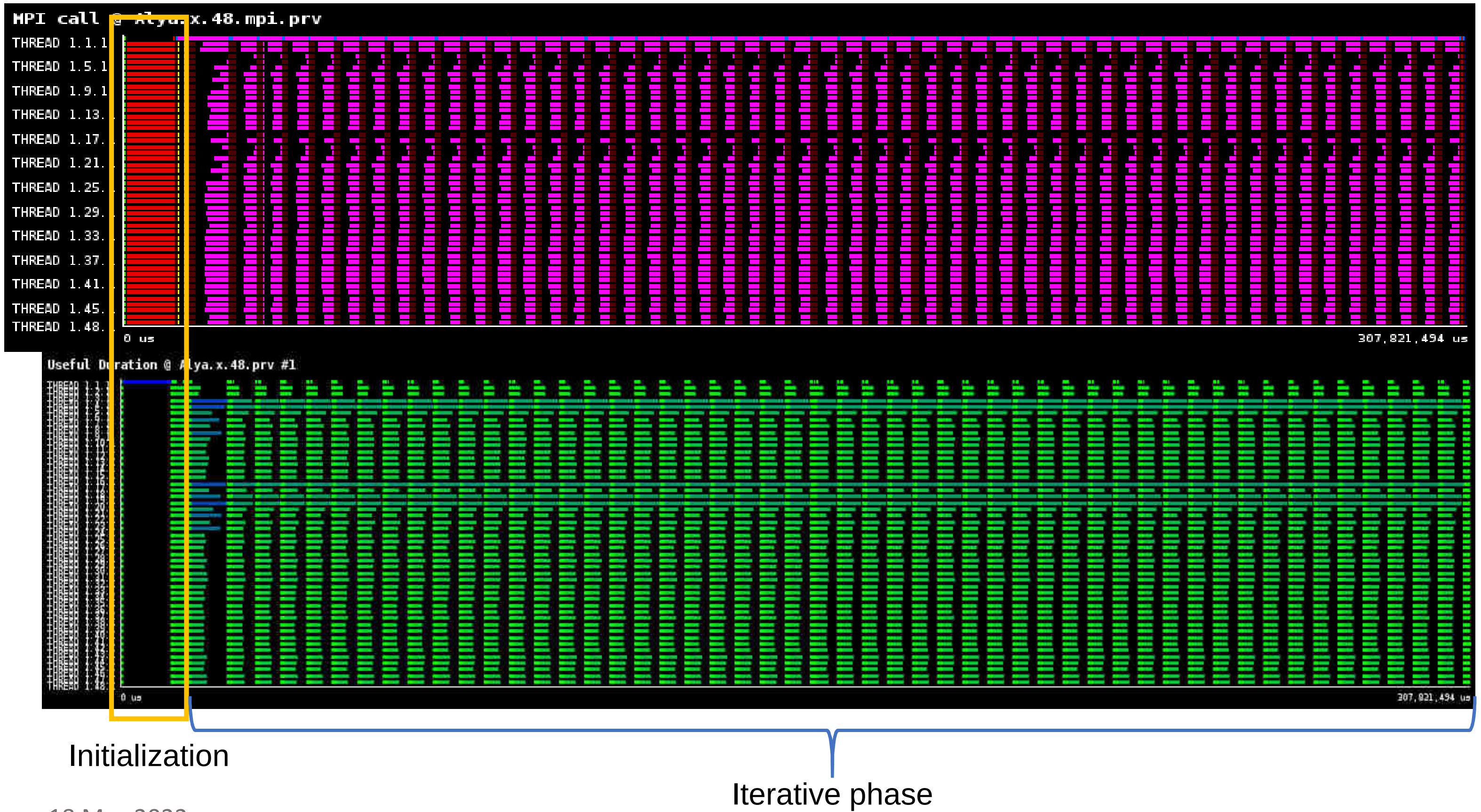
Identify structure

- Objectives:
  - Understand general structure
  - Identify initialization/finalization phases
  - Detect iterative pattern
  - Granularity

- Different levels of "difficulty" → Different levels of knowledge

Identify structure

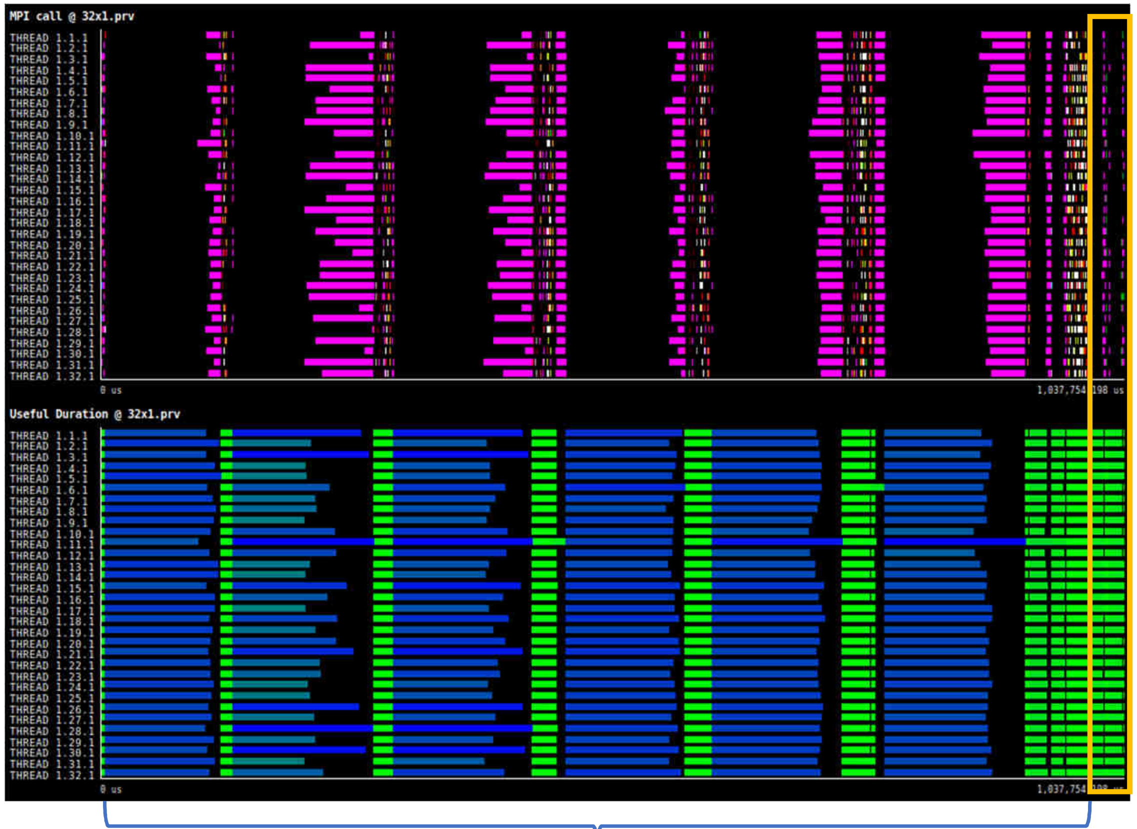
- Usually use “MPI calls” view or “Useful Duration”



- Clear iterative pattern
- With an initialization phase
- All iterations are similar
  - We can select a few to analyze

Identify structure

- Not always easy



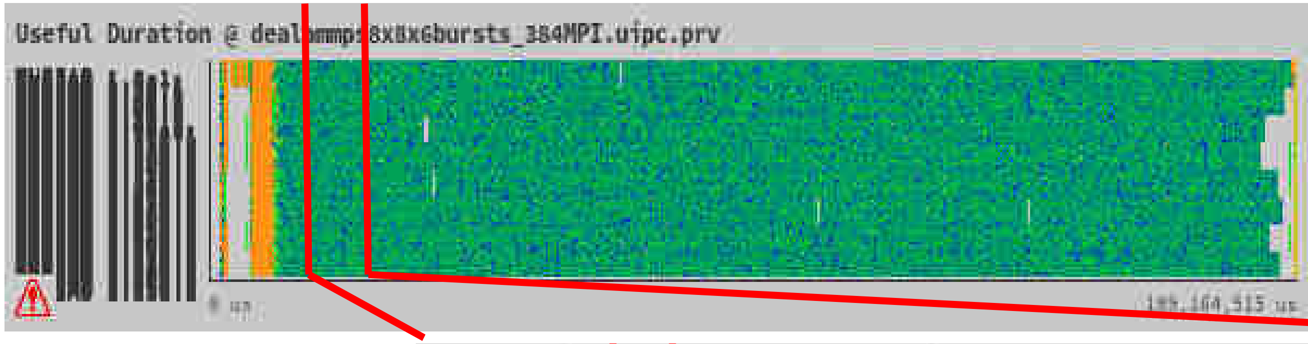
6 iterations

Finalization

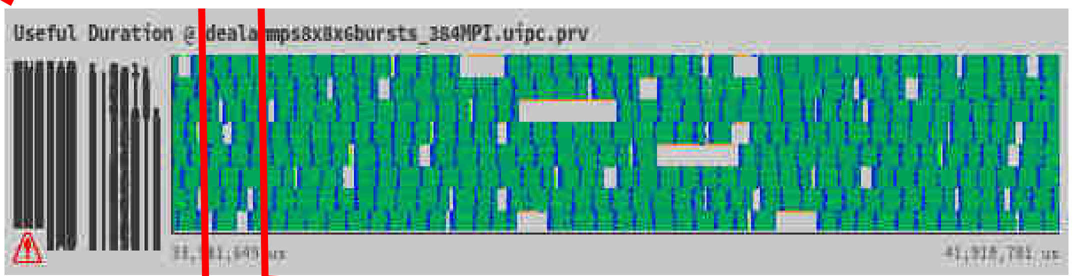
- There are 6 iterations and one finalization phase
- Iterations are not regular along time
  - Different pattern of load balance
  - Different pattern of durations

Identify structure

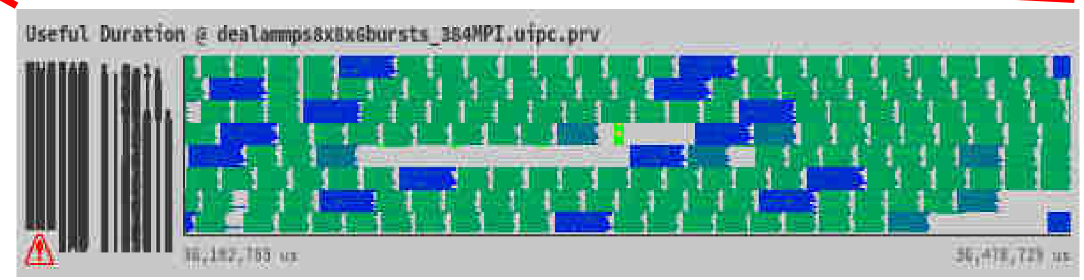
- Not always easy



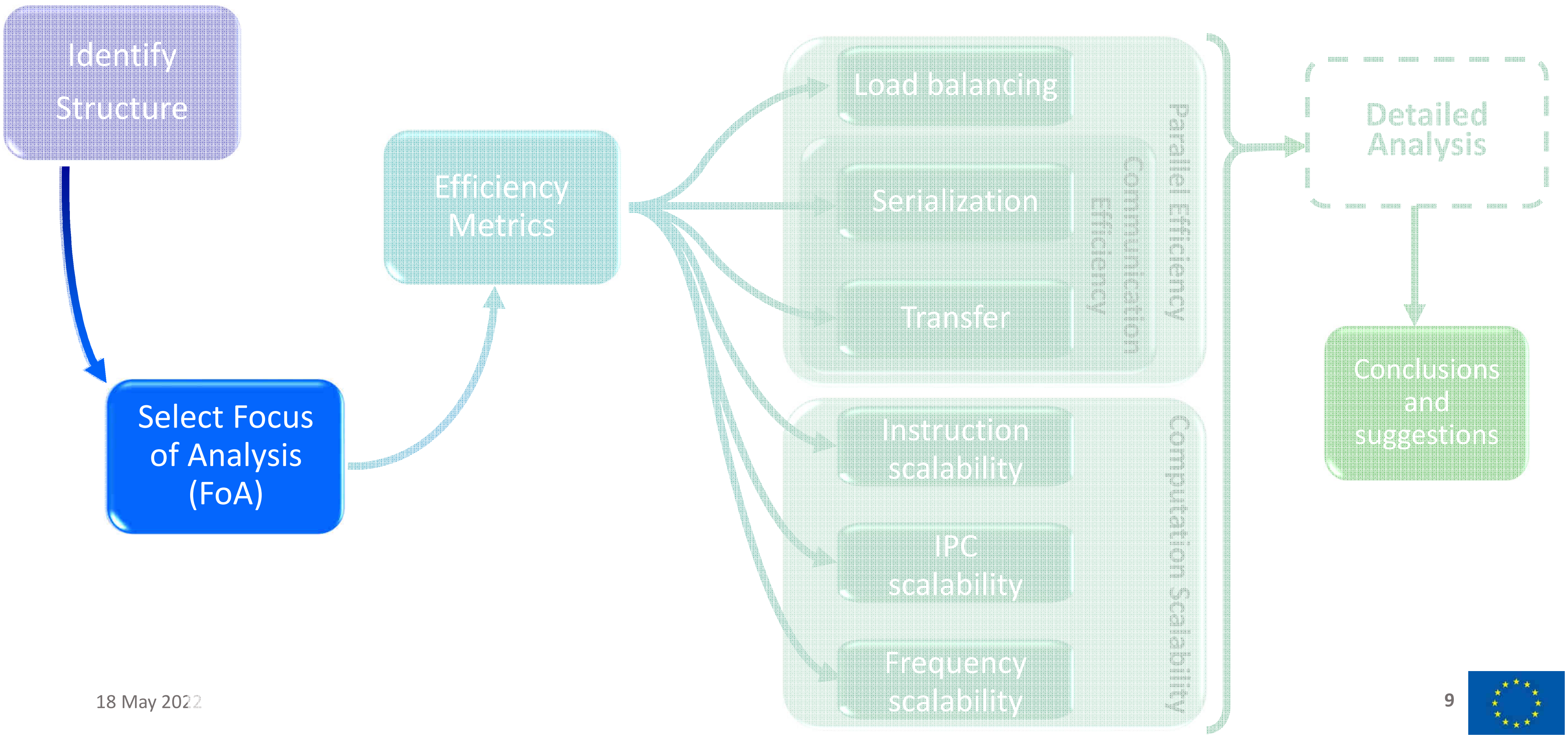
- It is not easy to detect an iterative structure
- No global synchronizations



- Zooming in we detect that synchronizations only happen at node level
- Not possible to determine iterations

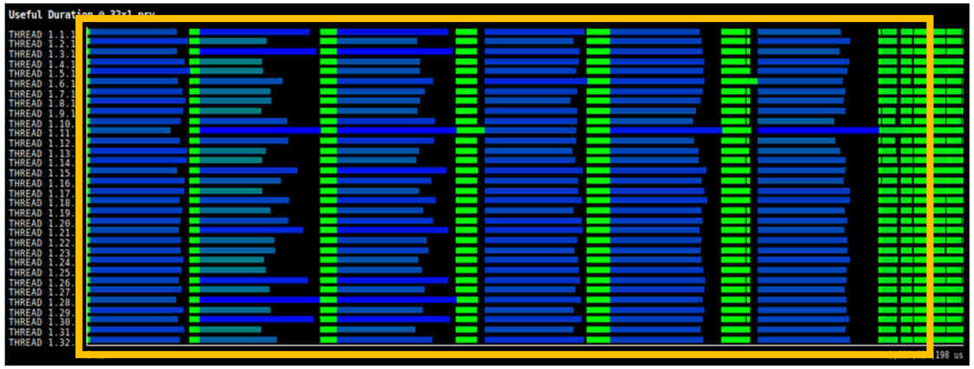


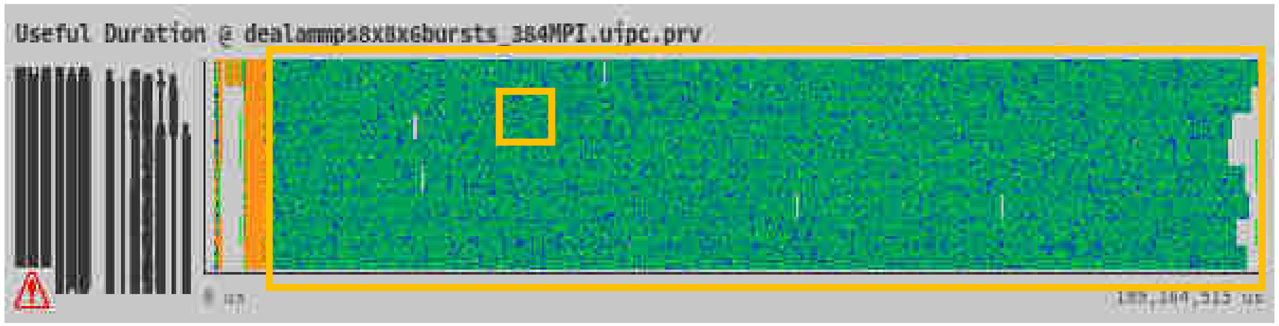
The journey


Identify Structure
Efficiency Metrics
Select Focus of Analysis (FoA)
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

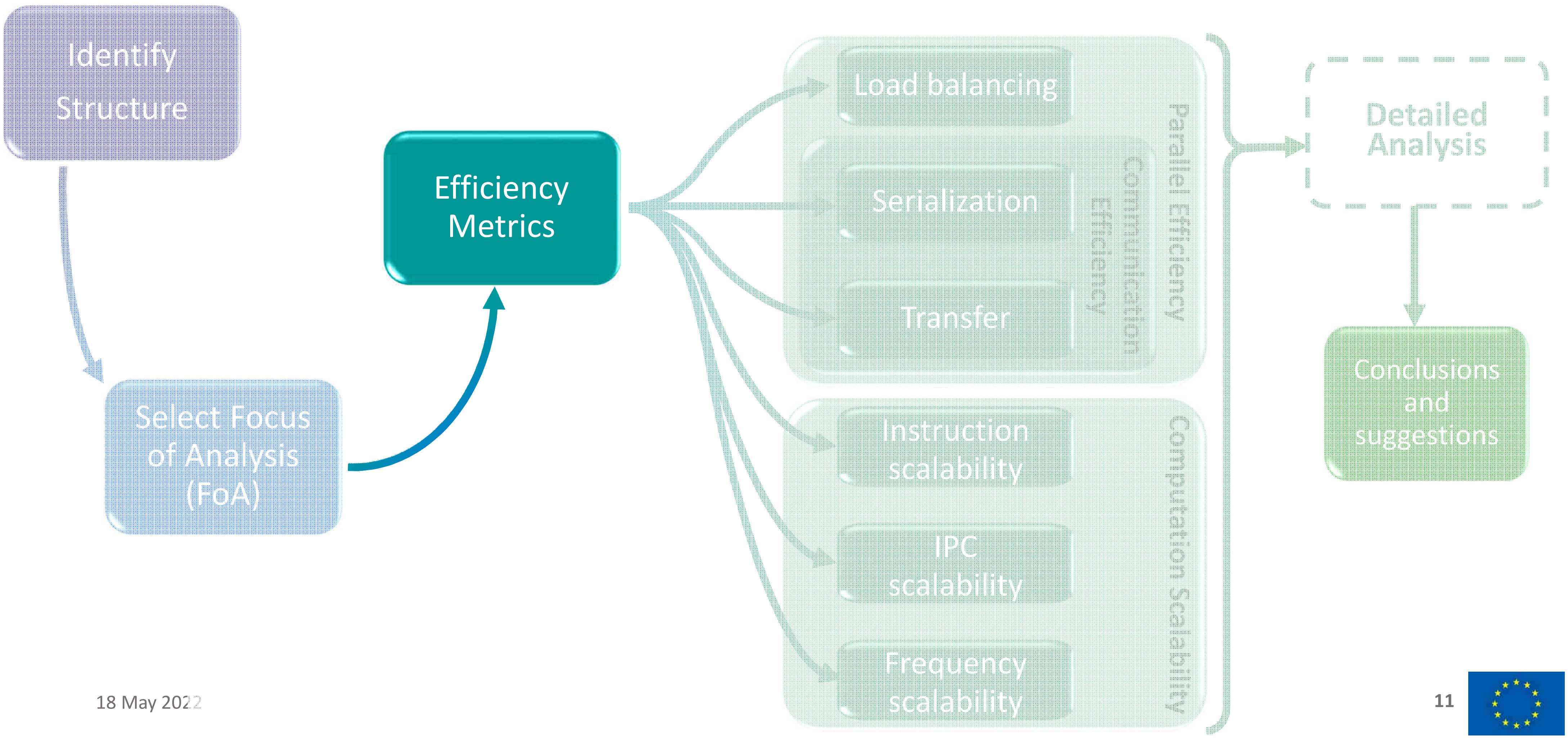
Select Focus of Analysis (FoA)

- **Objective:** Select the region we want to analyze
  - Not a correct answer, depends on the context of the analysis
  - For the same trace we may select two different FoA to perform two different studies with two different objectives.



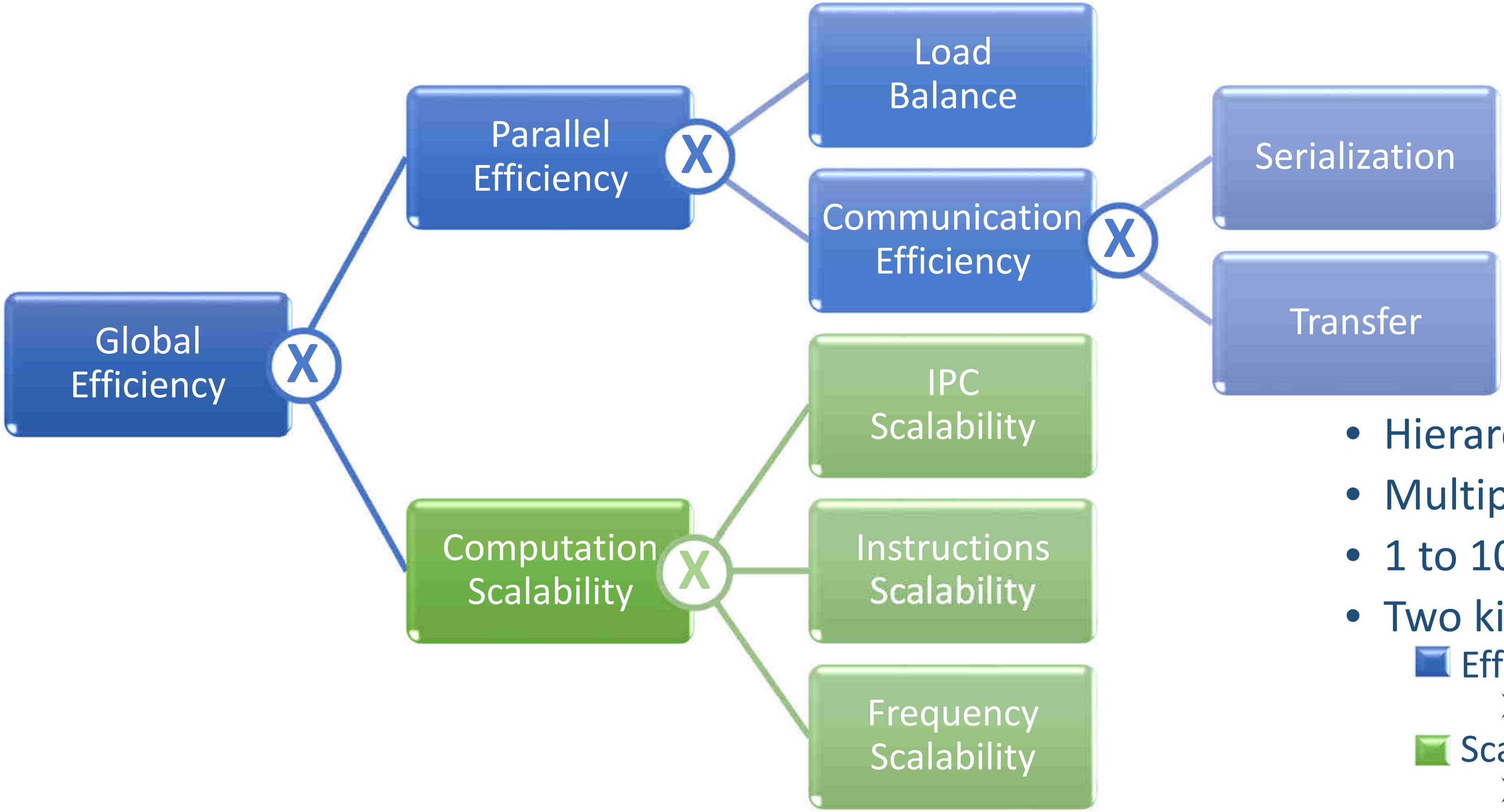


The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

The Efficiency Metrics



- Hierarchical model
- Multiplicative metrics
- 1 to 100% scale
- Two kind of metrics:
  - Efficiency metrics
    - Absolute metrics
  - Scalability metrics
    - Relative to a base case
    - 100% for the base case

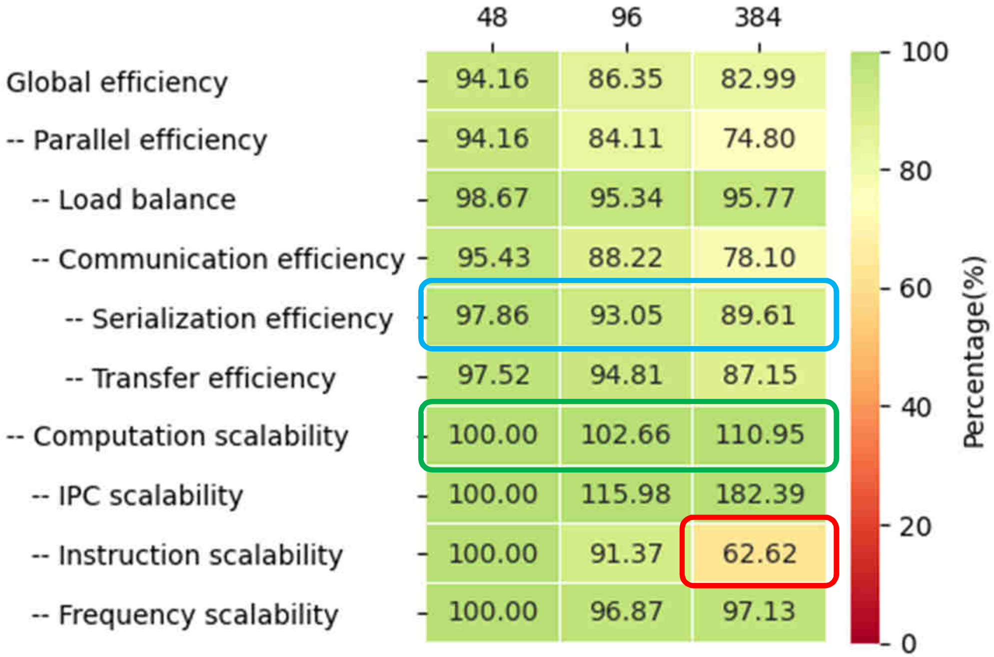
The Efficiency Metrics

- What they are NOT
  - The end of the journey
- What they are…
  - … a general mechanism to describe the fundamental concepts of parallelism
  - … a hint that tells where to look
  - … a way to quantify efficiency loss
  - … a fair comparison between different…
    - … code versions
    - … architectures
    - … core counts (scalability)
    - … applications

- For each metric we are going to see:
  - What it quantifies
  - How it is computed
    - Formula
    - Graphical
  - Interpretation
  - Where to look next

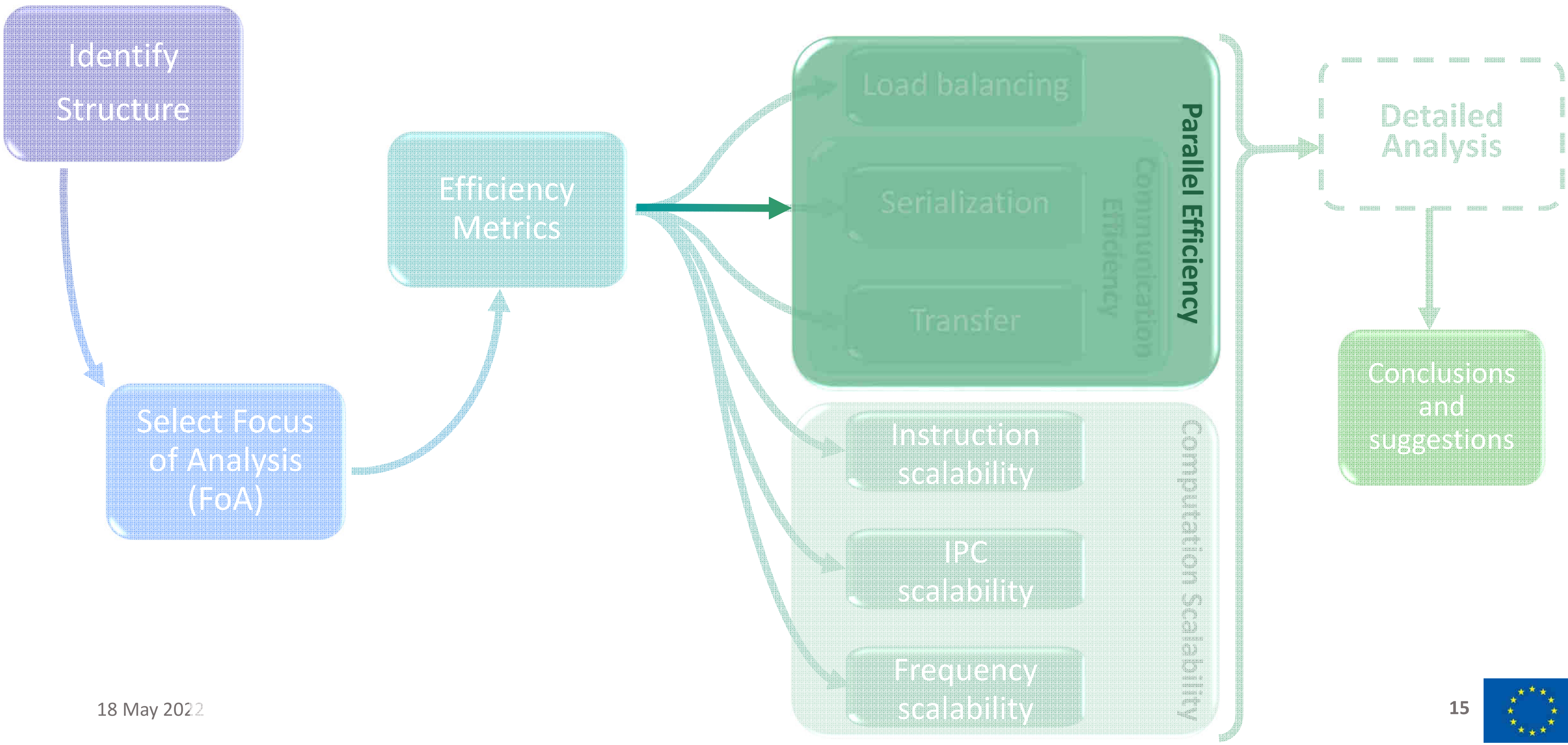
The Efficiency Metrics

- Usually shown in a table
  - Rows: Metrics
  - Columns: Different traces
- With colored cells as a heat map
- What to look for?
  - Low values
  - Trend
  - High values



	48	96	384
Global efficiency	94.16	86.35	82.99
-- Parallel efficiency	94.16	84.11	74.80
-- Load balance	98.67	95.34	95.77
-- Communication efficiency	95.43	88.22	78.10
-- Serialization efficiency	97.86	93.05	89.61
-- Transfer efficiency	97.52	94.81	87.15
-- Computation scalability	100.00	102.66	110.95
-- IPC scalability	100.00	115.98	182.39
-- Instruction scalability	100.00	91.37	62.62
-- Frequency scalability	100.00	96.87	97.13

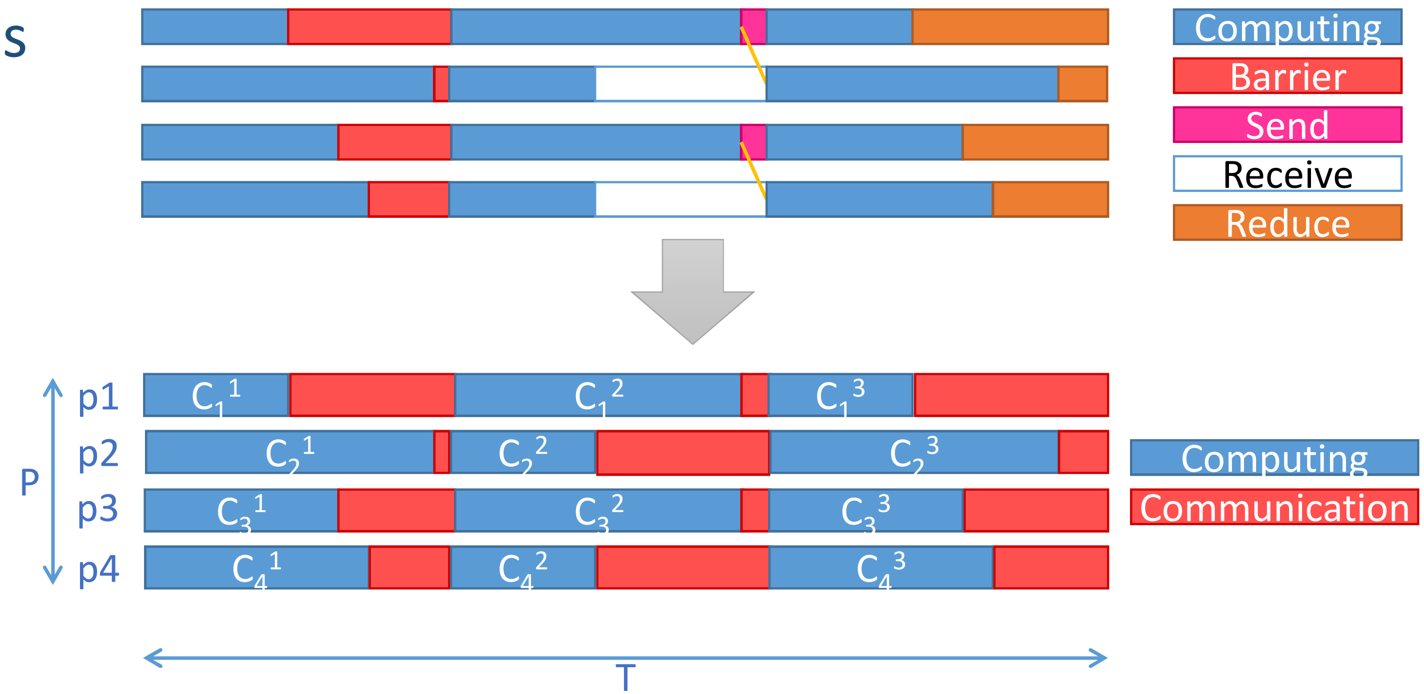
The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Some semantics first

- The state of a processes is simplified to two values:
  - Useful == Computing
  - Not useful == Otherwise



- T = Elapsed time
- P = Number of processes
- c_i = Compute time of process i
  - $c_i = \sum_{j=1}^{n} c_i{}^j$
- C = Total compute time
  - $C = \sum_{i=1}^{P} c_i$

Parallel Efficiency

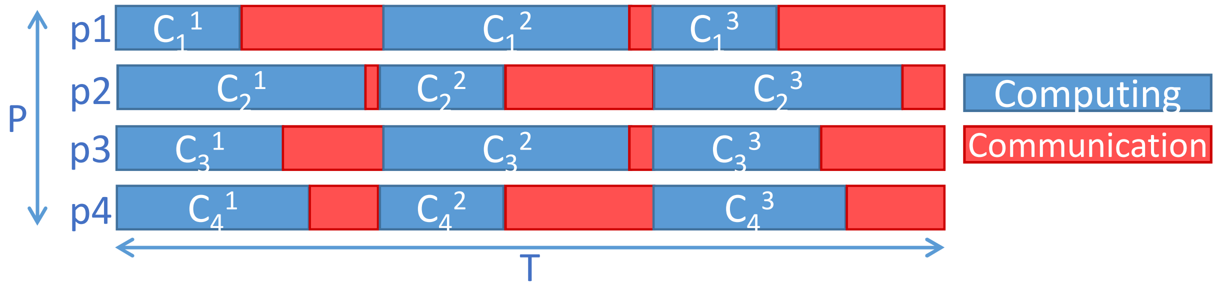
Quantifies: The extent to which all resources in the system are kept active doing useful work

How it is computed: Ratio between time used to do useful computation and consumed cpu time

Parallel Efficiency = ▭ / (▭ + ▭)

How it is computed:

$$Parallel\ Efficiency = \frac{\sum_{i=1}^{P} c_i}{P * T}$$

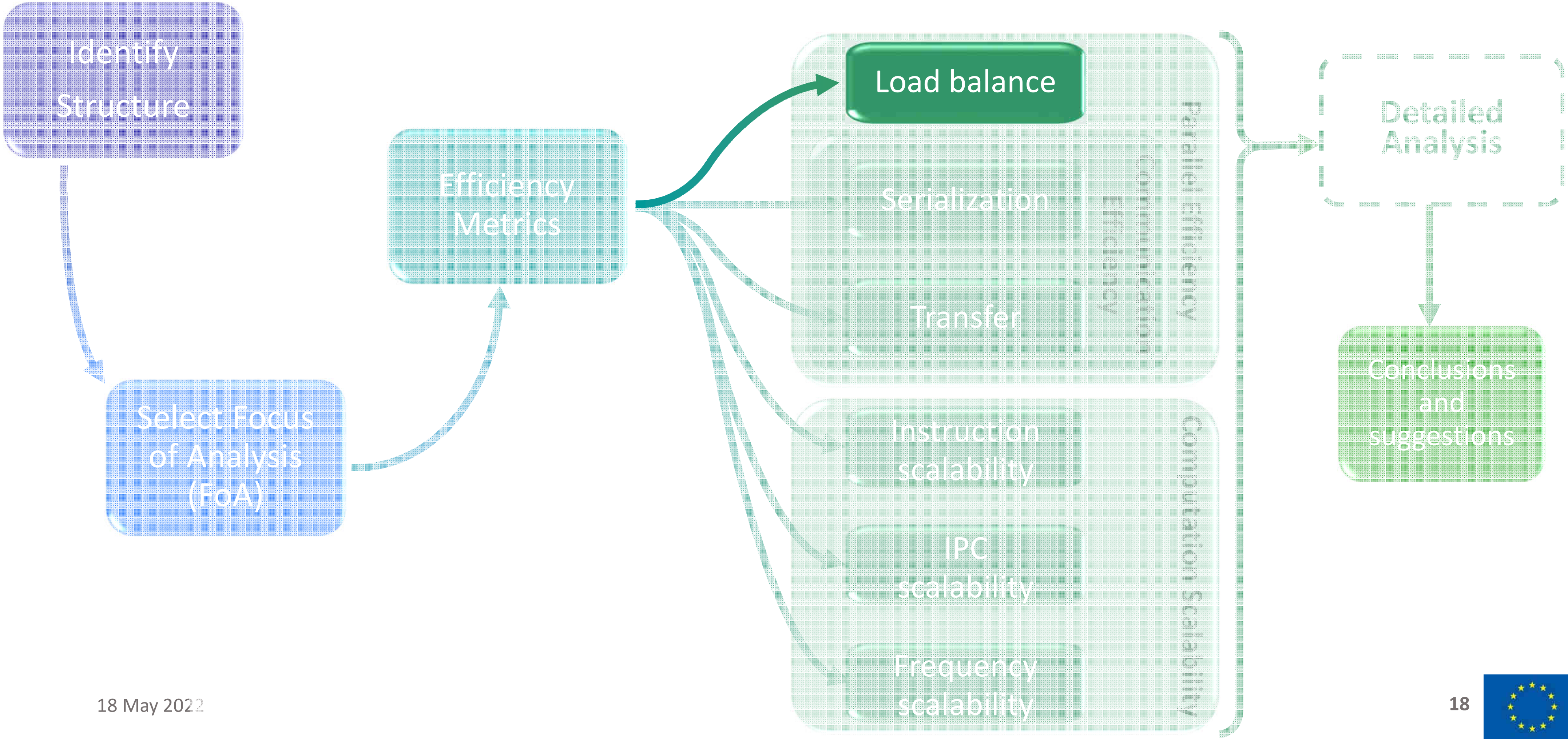


Interpretation: A low value indicates that a low fraction of the time consumed is used to do useful computation.

Where to look next?

- Look at its child metrics

The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balance
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Load Balance

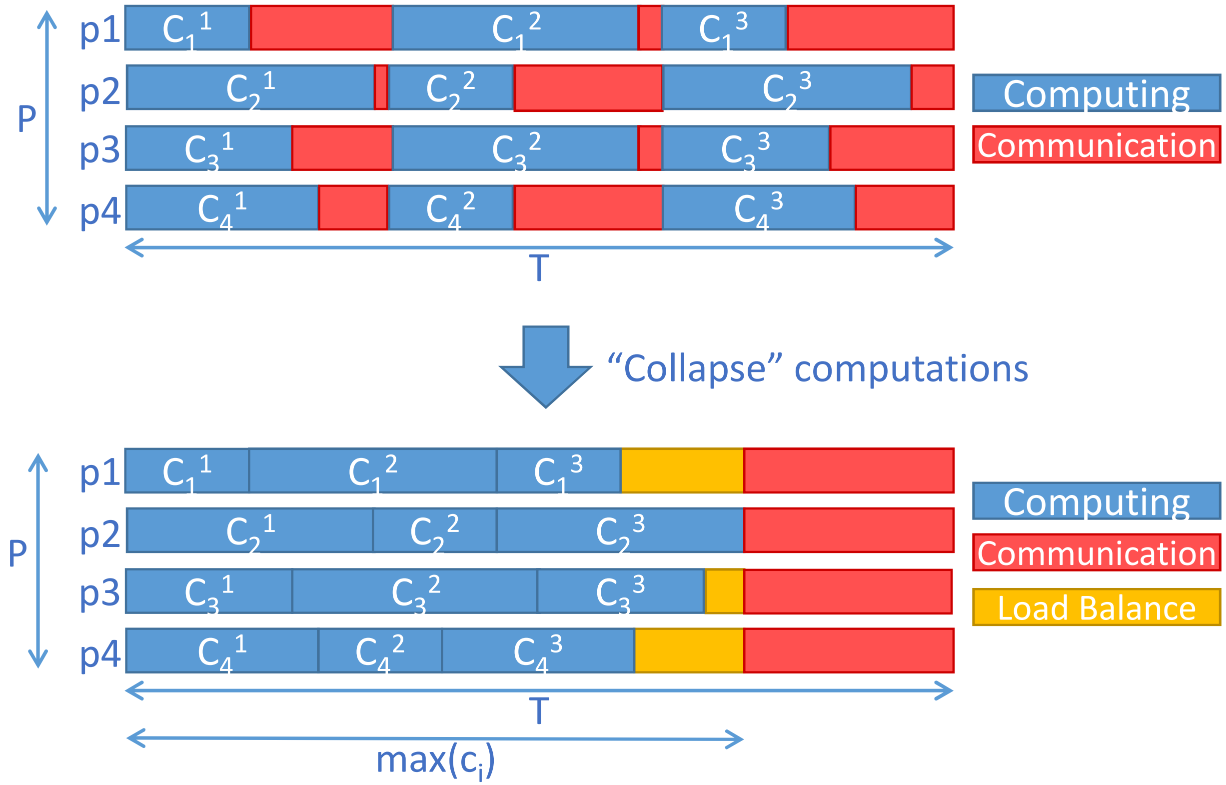
Quantifies: The efficiency loss due to the global distribution of work among processes.

How it is computed: Ratio between time used to do useful computation and the useful computation of the most loaded process multiplied by the number of processes

Load Balance = [blue] / ([blue] + [yellow])

How it is computed:

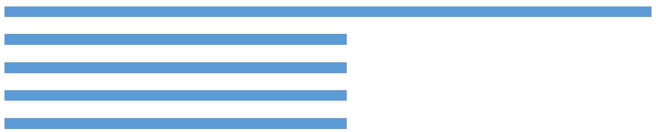
$$Load\ Balance = \frac{\sum_{i=1}^{P} C_i}{max^{P}{}_{i=1}(c_i) * P}$$



Load Balance

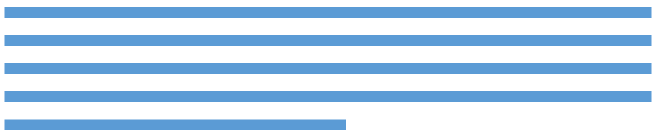
Interpretation: A low value in this metric indicates that more highly loaded processes keep other processes idle for a significant amount of time.

- A single highly loaded process will make this metric report a low value



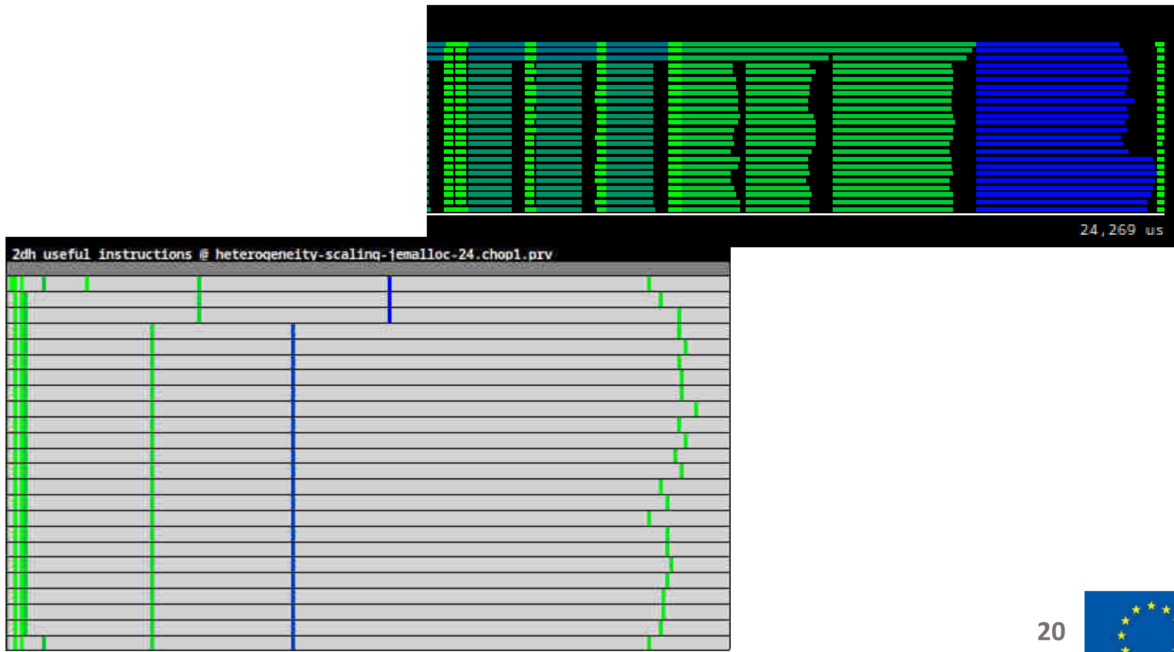
$$LB = \frac{2+1+1+1+1}{2*5} = \frac{6}{10} = 0.6$$

- A single low loaded process won't have an effect on this metric

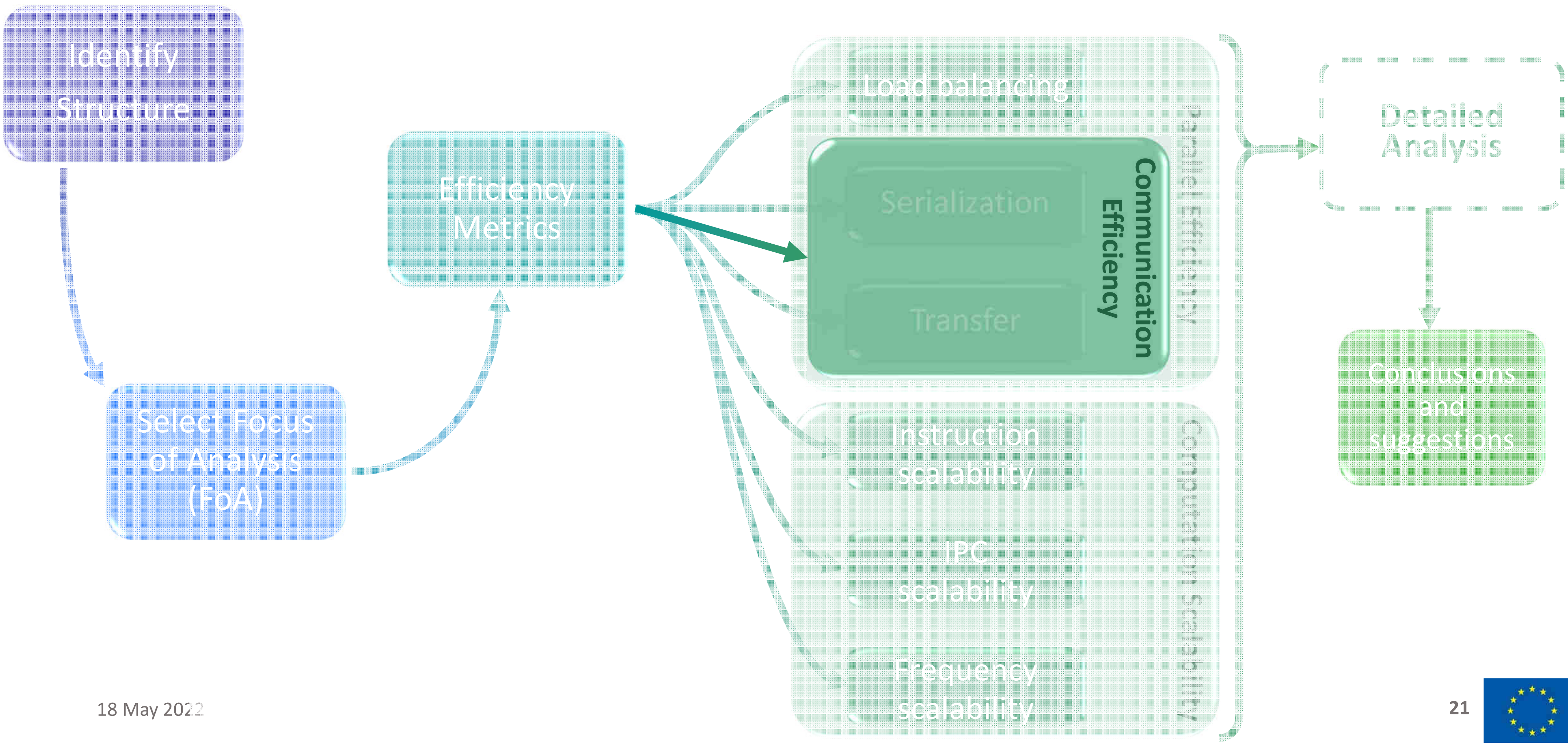


$$LB = \frac{2+2+2+2+1}{2*5} = \frac{9}{10} = 0.9$$

- What to look next?
  - We try to understand the cause of the Load Imbalance
  - In general can have 3 sources
    - Number of instructions → Histogram of useful instructions
    - IPC → Histogram of IPC
    - Frequency → Histogram of cycles per us



The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Communication Efficiency

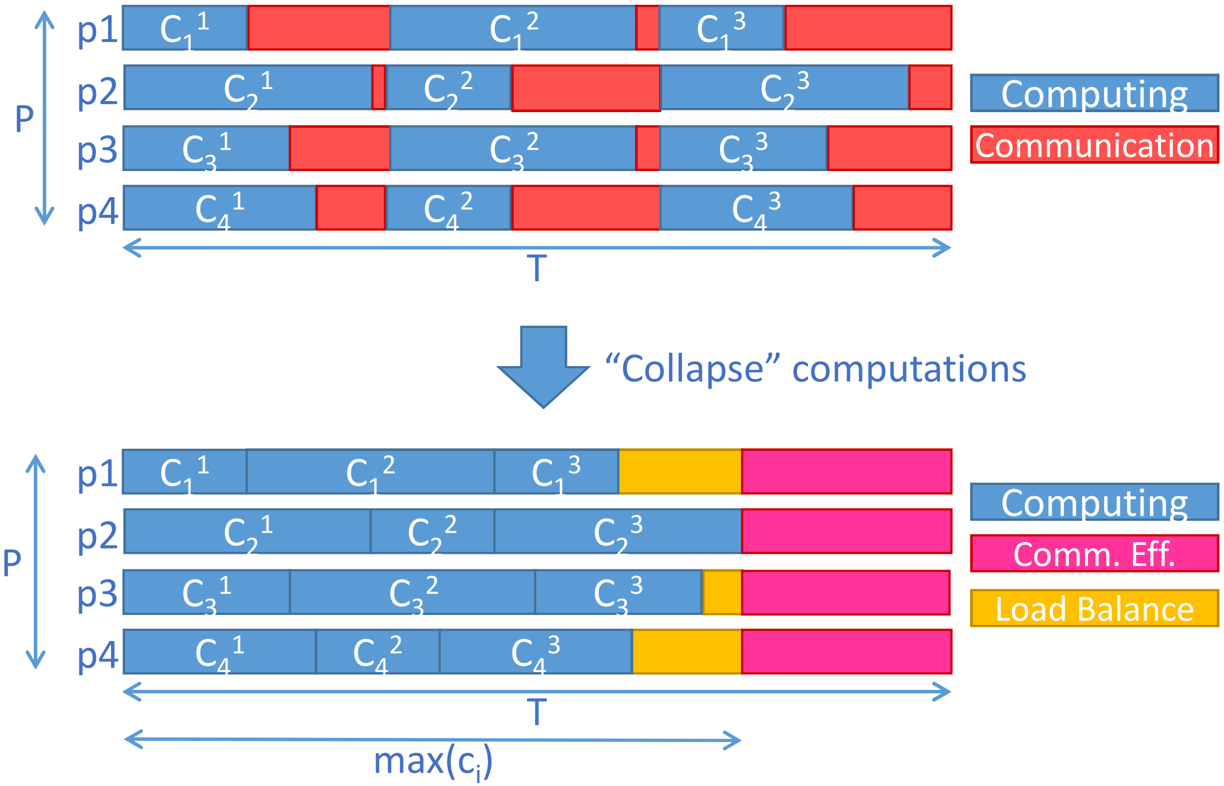
Quantifies: The efficiency loss due to the communication of data. Be it due to synchronizations between processes or to the overhead introduced by the communication itself. Excluding time loss due to global load imbalance

How it is computed: Ratio between the useful computation time of the most loaded processes and the total elapsed time

Communication Eff.= (+) / (+ +)

How it is computed:

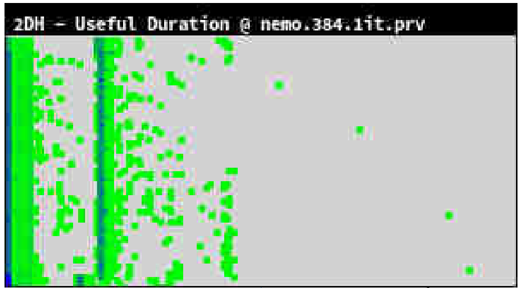
$$Communication\ Eff. = \frac{max^{P}{}_{i=1}(c_i)}{T}$$

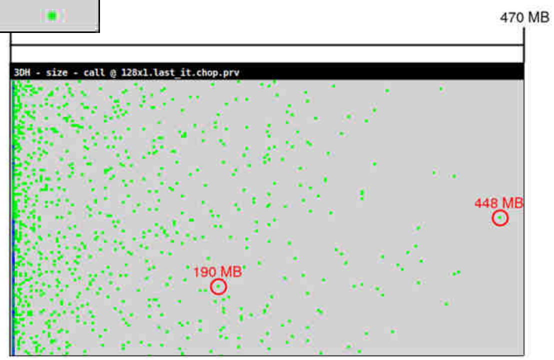


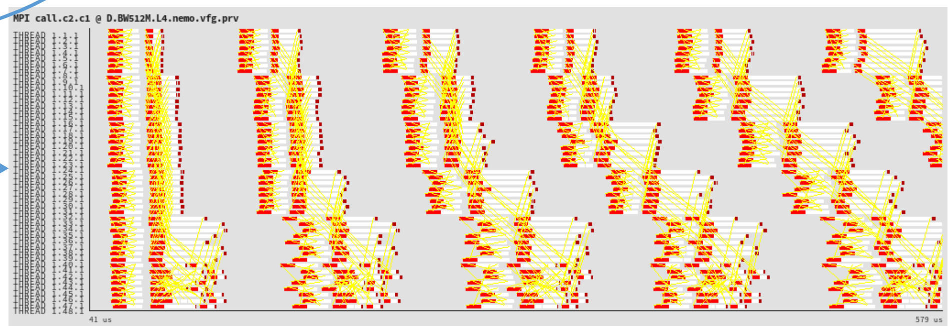
Communication Efficiency

Interpretation: A low value in this metric indicates that the interaction between processes is impacting the performance.

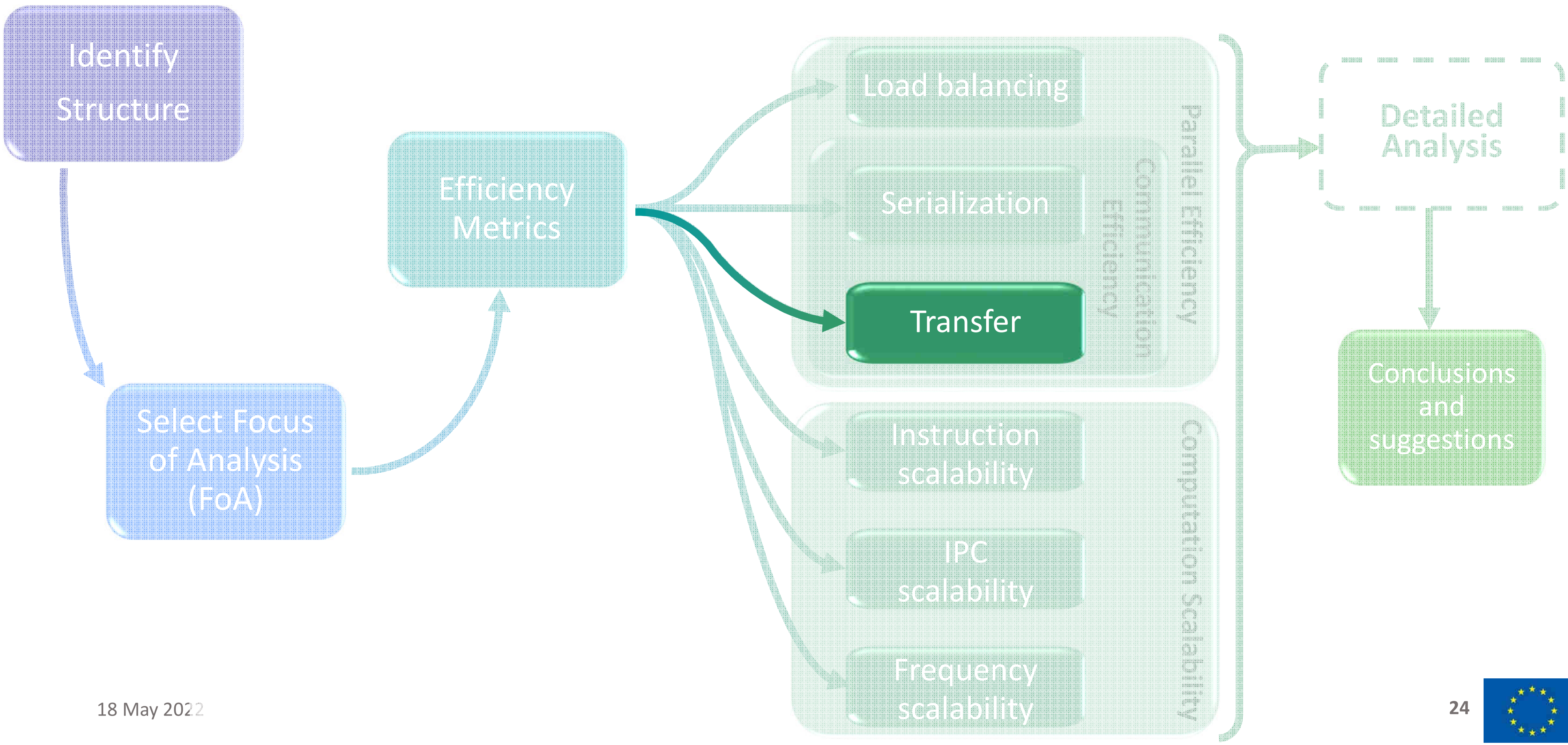
- This metric can report good values in codes where a profile reports significant time in MPI and that would be reported by a bad load balance efficiency
- What to look next?
- If possible child metrics
- If not…
  - How many MPI calls are made?
    - Histogram of MPI calls
  - How often? Granularity of computations
    - Useful duration
  - How much data is sent?
    - Bytes sent per MPI call
  - Which are the semantics of the communication?
    - Chains of dependences?
      - MPI calls







The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

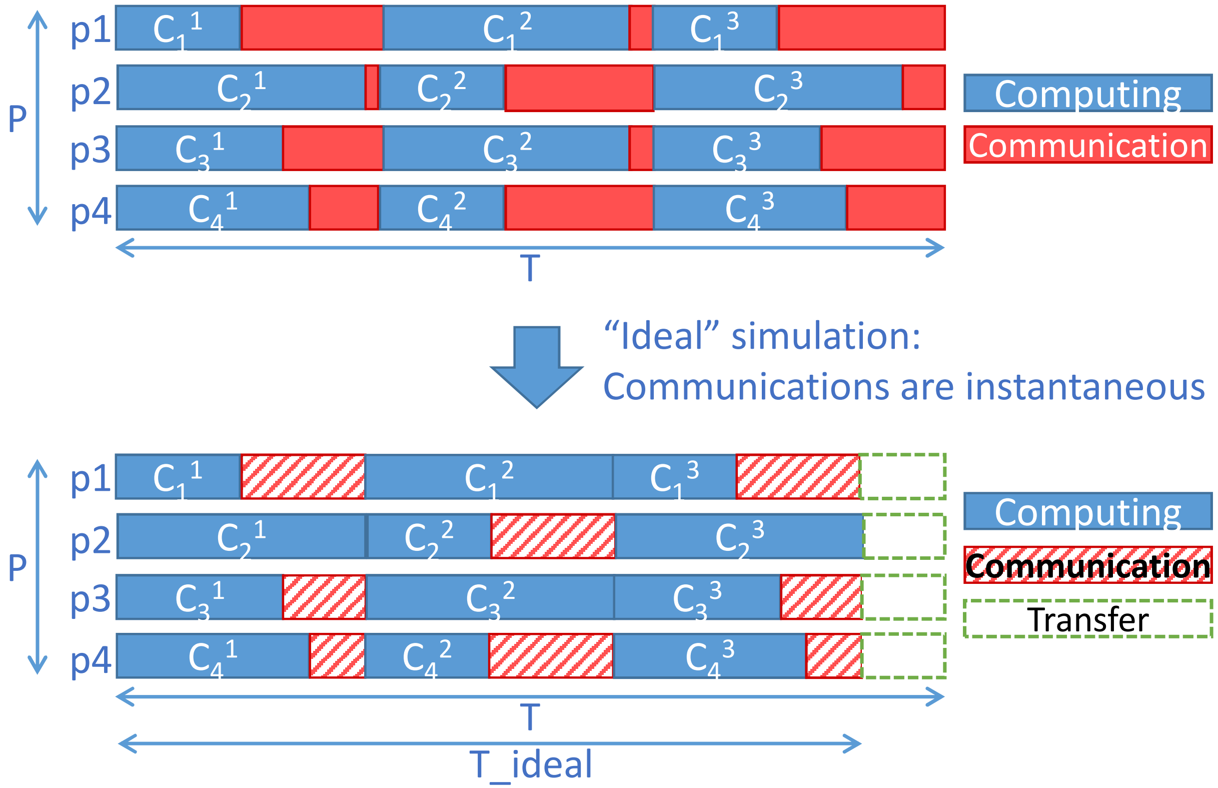
Transfer Efficiency

Quantifies: Efficiency loss related to the non instantaneous nature of communication mechanisms. Includes time to transmit the data over the physical channel and the overhead in the libraries.

How it is computed: Ratio between elapsed time in the ideal simulation and the elapsed time in the real execution

How it is computed:

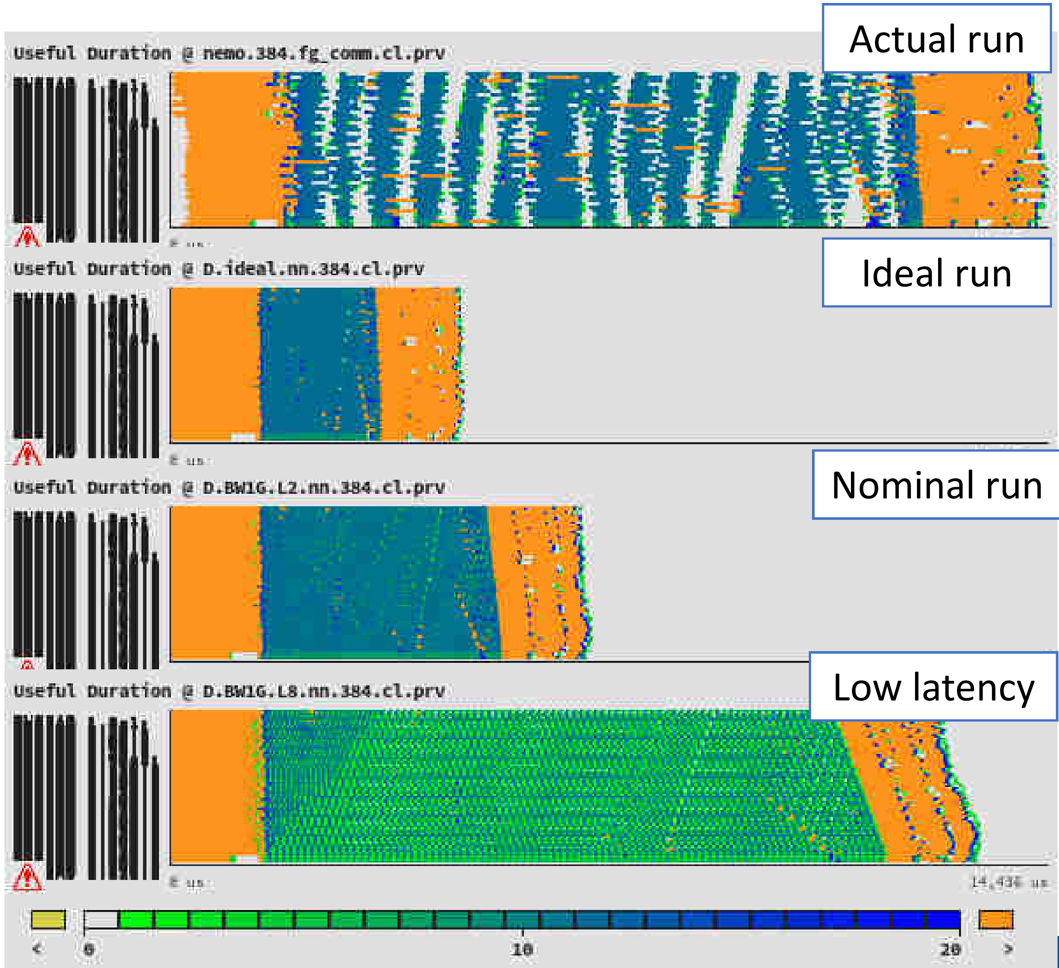
$$Transfer\ Eff. = \frac{T_{ideal}}{T}$$



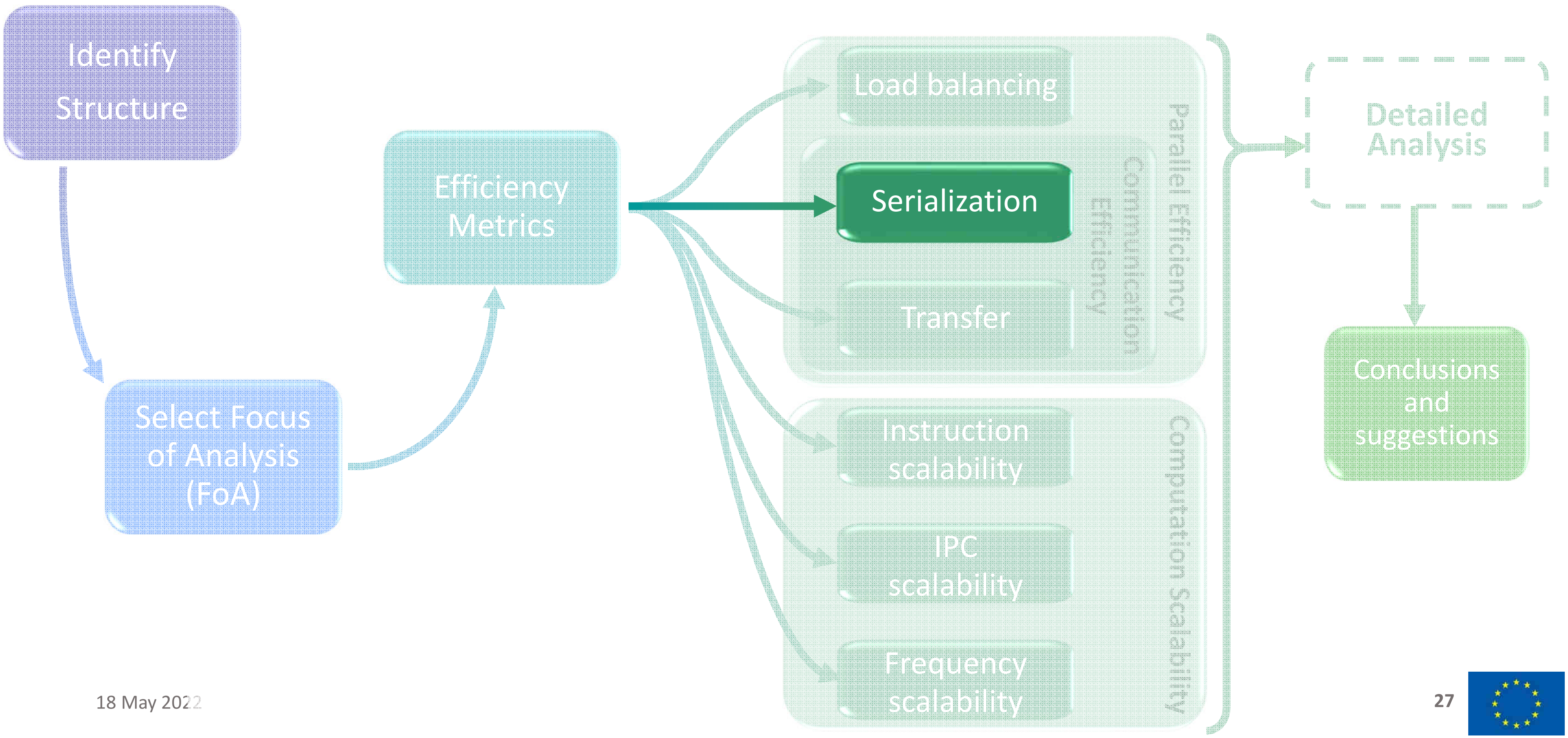
Transfer Efficiency

Interpretation: Low values indicate that the execution is suffering from a high overhead of the runtime or a poor latency or bandwidth of the network.

- What to look next?
  - Determine if the transfer problem is Bandwidth or Latency
    - ➢ Note that in Latency we include the overhead of the communication library
    - ➢ Use Simulations of Dimemas
      - o Different BW and latency



The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Serialization Efficiency

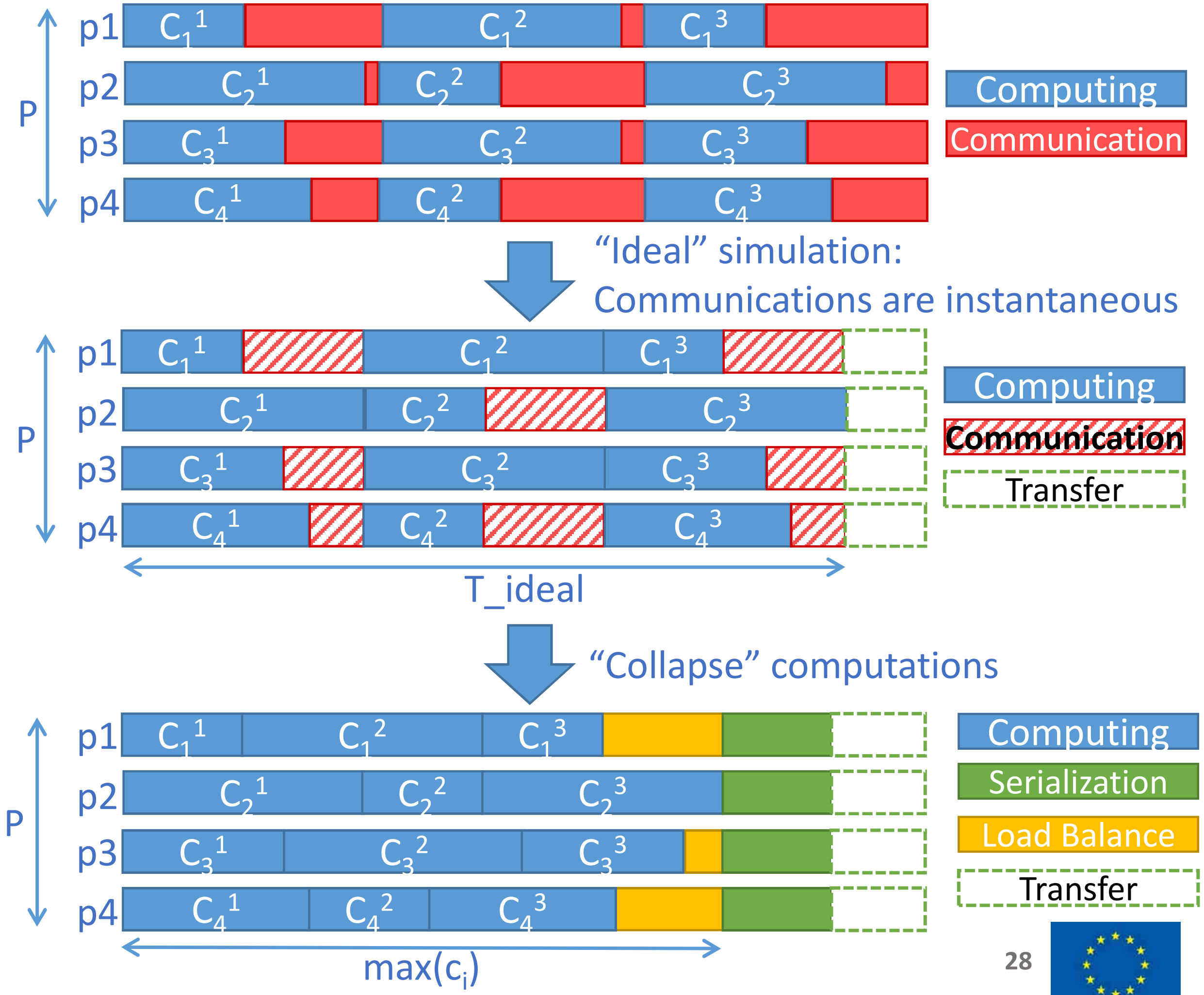
Quantifies: Inefficiencies caused by circular dependences or non-uniform imbalances

How it is computed: Ratio between useful computation of most loaded process and elapsed time in the ideal simulation

Serialization Eff.= (+) / (+)

How it is computed:

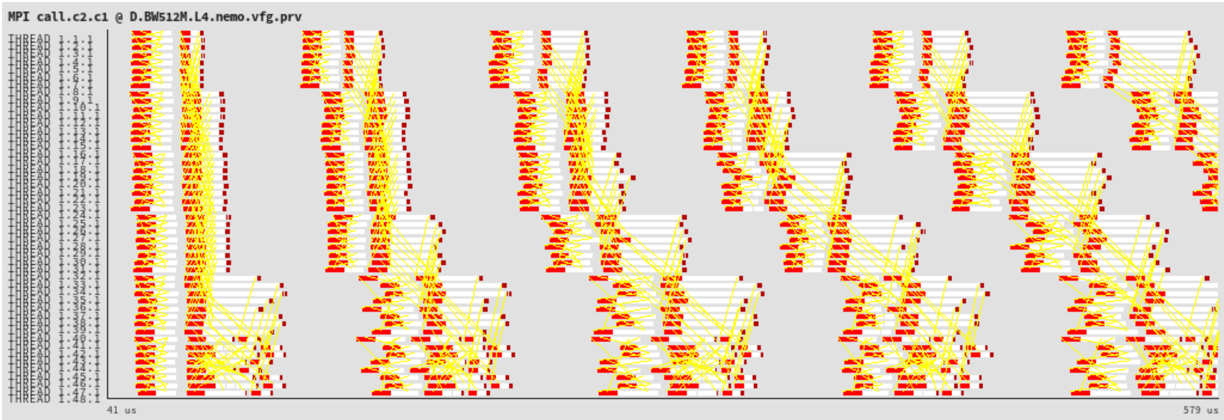
$$Serialization\ Eff. = \frac{max^{P}{}_{i=1}(c_i)}{T_{ideal}}$$



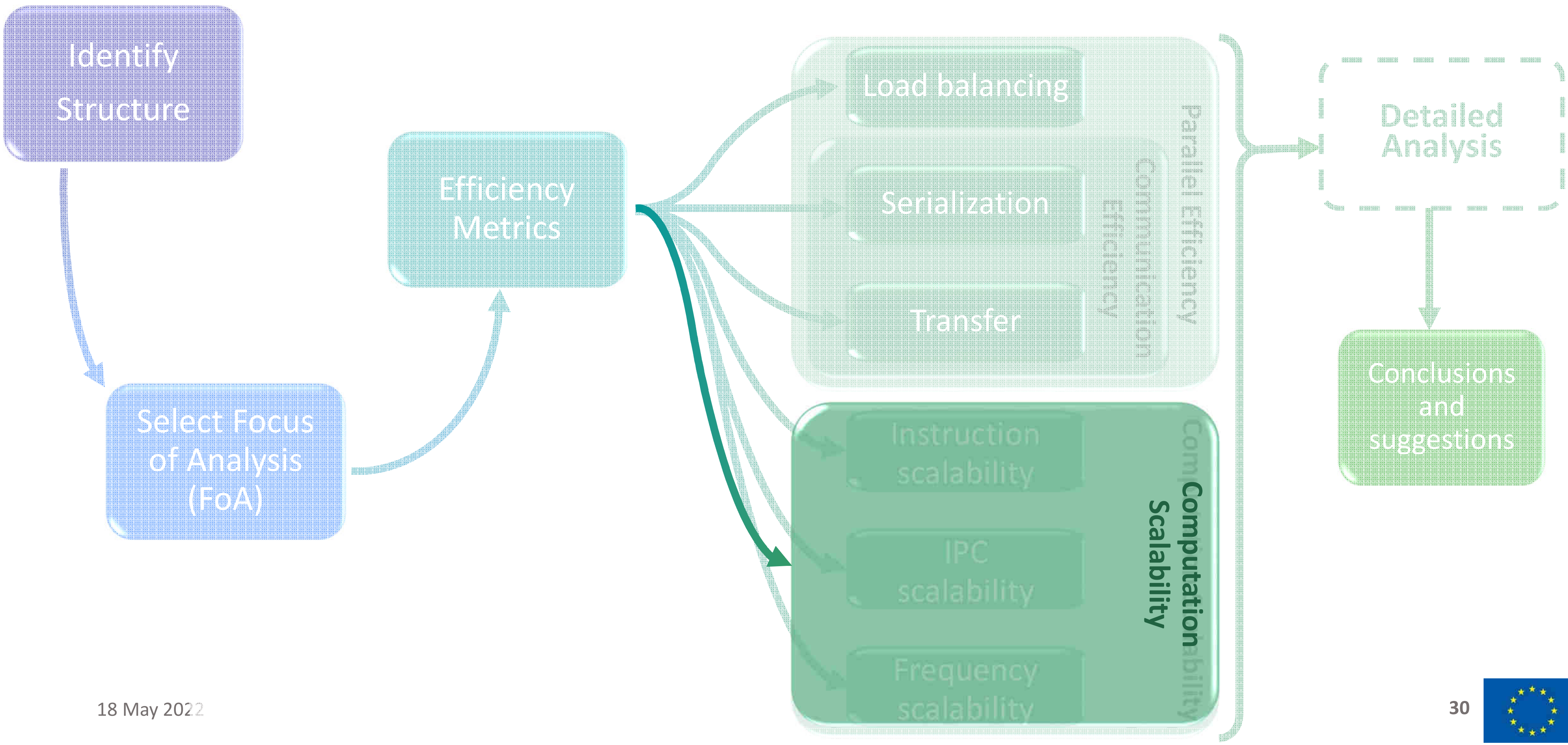
Serialization Efficiency

Interpretation: A low value indicates the existence of circular dependences. They can be caused by actual algorithmic serialization, irregularities in the load of processes during the execution or noise.

- What to look next?
  - Discard noise
    - Cycles per us
  - Try to identify causes for circular waits.
  - Understand semantics of the communication
    - MPI Calls



The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Computation Scalability

Quantifies: How the time spent computing scales with respect to the reference case.

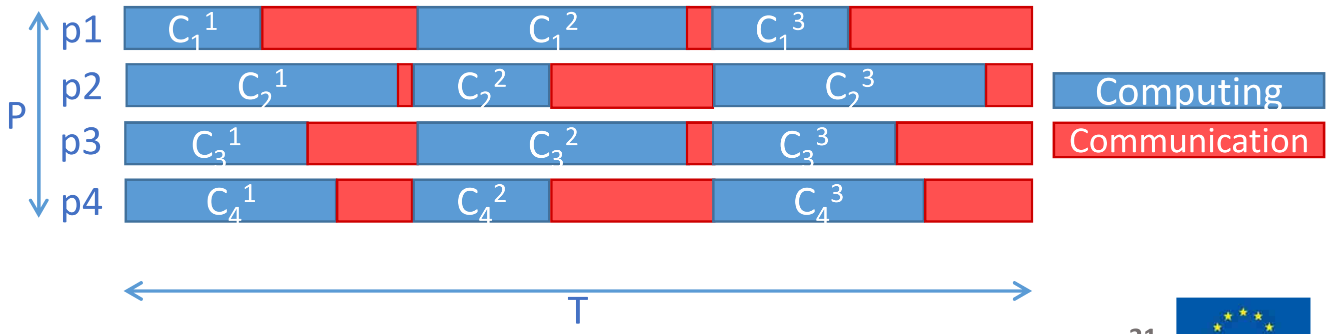
How it is computed: Ratio between time spent on useful computation in the reference case and the time spent on useful computation on current run.

How it is computed:

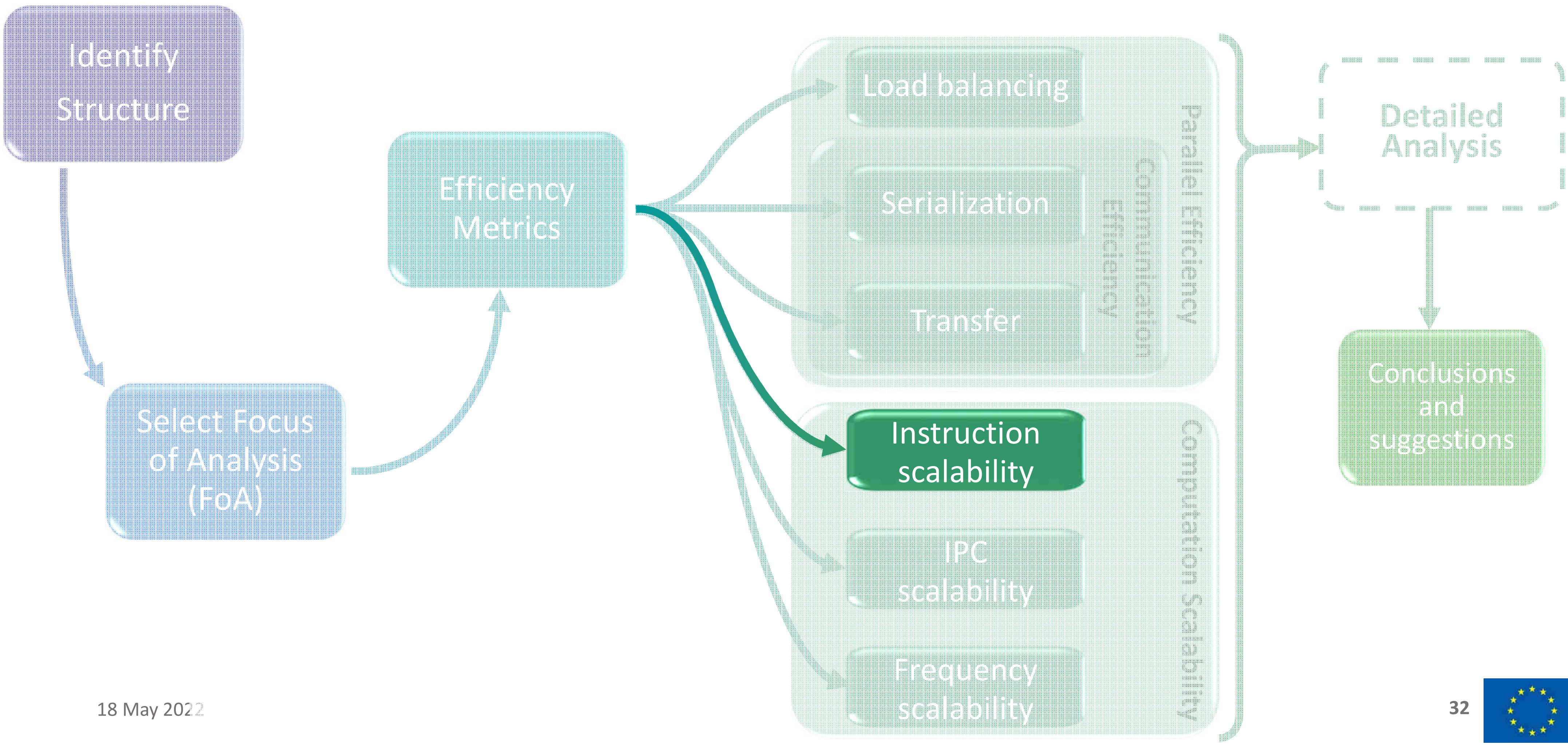
$$Computation\ Sc. = \frac{C_{ref}}{C_{current}}$$

Interpretation: A low value indicates time grows per core count. Ideally the total compute time to solve a problem should be constant, independent of core count (in strong scaling).

- Relative metric, based on a reference case.
  - C_{ref} = Useful Computation of the reference run
  - $C_{current}$ = Useful computation current run
- 100% for the reference case
- Can be weak or strong scaling
  - Strong scaling for all the formulas presented
    - In strong scaling we assume total compute time and total number of instructions should remain constant as we increase the number of processes
  - In an analogous way weak scaling can be computed
- High level metric composed by three child metrics based on:
  - T = # instr. / (IPC * freq)



The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Instructions Scalability

Quantifies: How the number of instructions scales with respect to the reference case.

How it is computed: Ratio between number of instructions executed on the reference case and the number of instructions executed on current run.

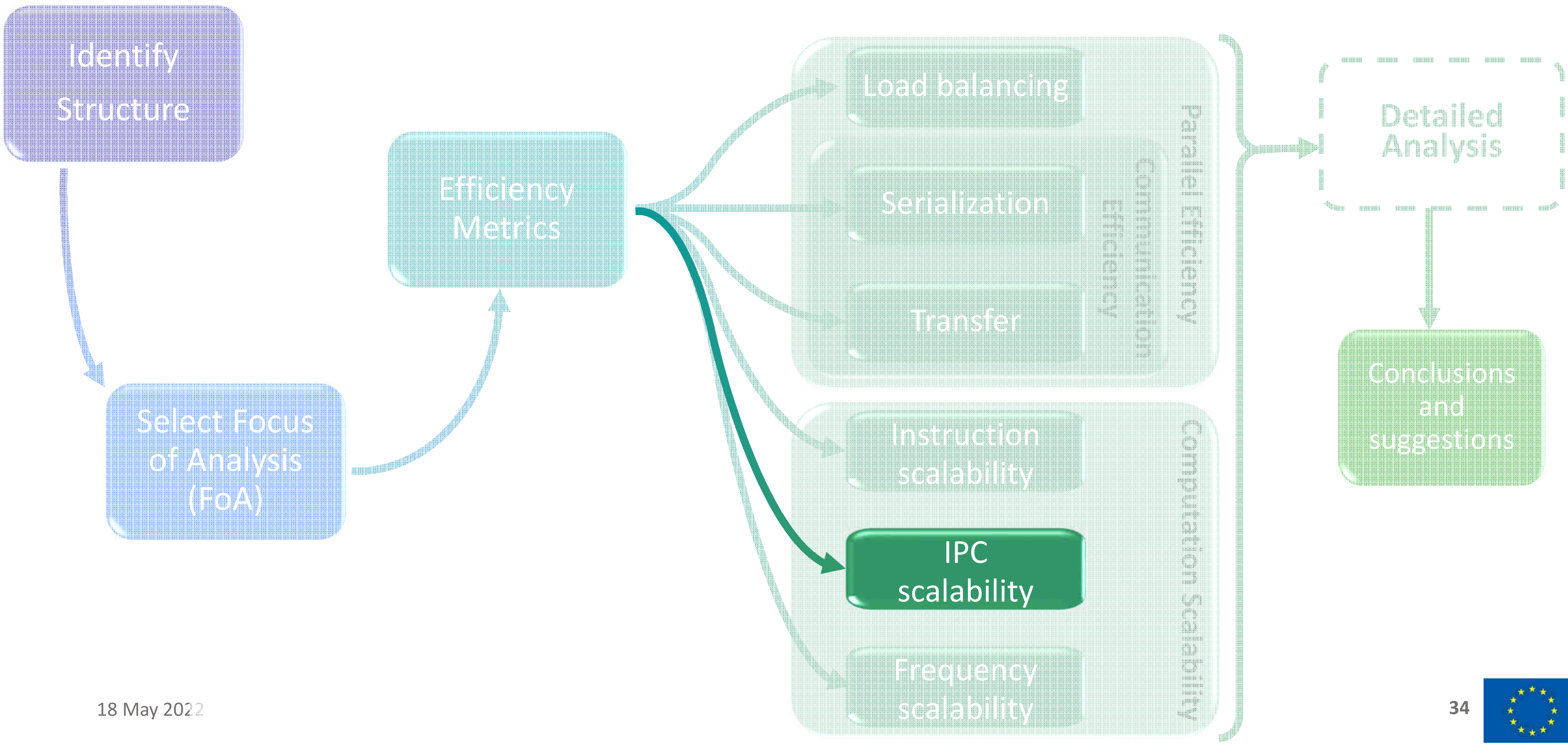
How it is computed:

$$Instructions\ Sc. = \frac{I_{ref}}{I_{current}}$$

- Relative metric, based on a reference case.
  - I_{ref} = Total number of instructions executed on the reference run
  - $I_{current}$ = Total number of instructions executed on the current run
- 100% for reference case

Interpretation: A value less than 100 indicate that the total number of instructions to solve the problem grows with core count, which ideally should not be the case.
May be caused by code replication (computed by all processes, by an increase in the surface to volume ratio when computations on the surface/boundary are "replicated", ...

The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

IPC Scalability

Quantifies: How the IPC scales with respect to the reference case.

How it is computed: Ratio between average IPC on the current run and the average IPC on the reference run.

How it is computed:

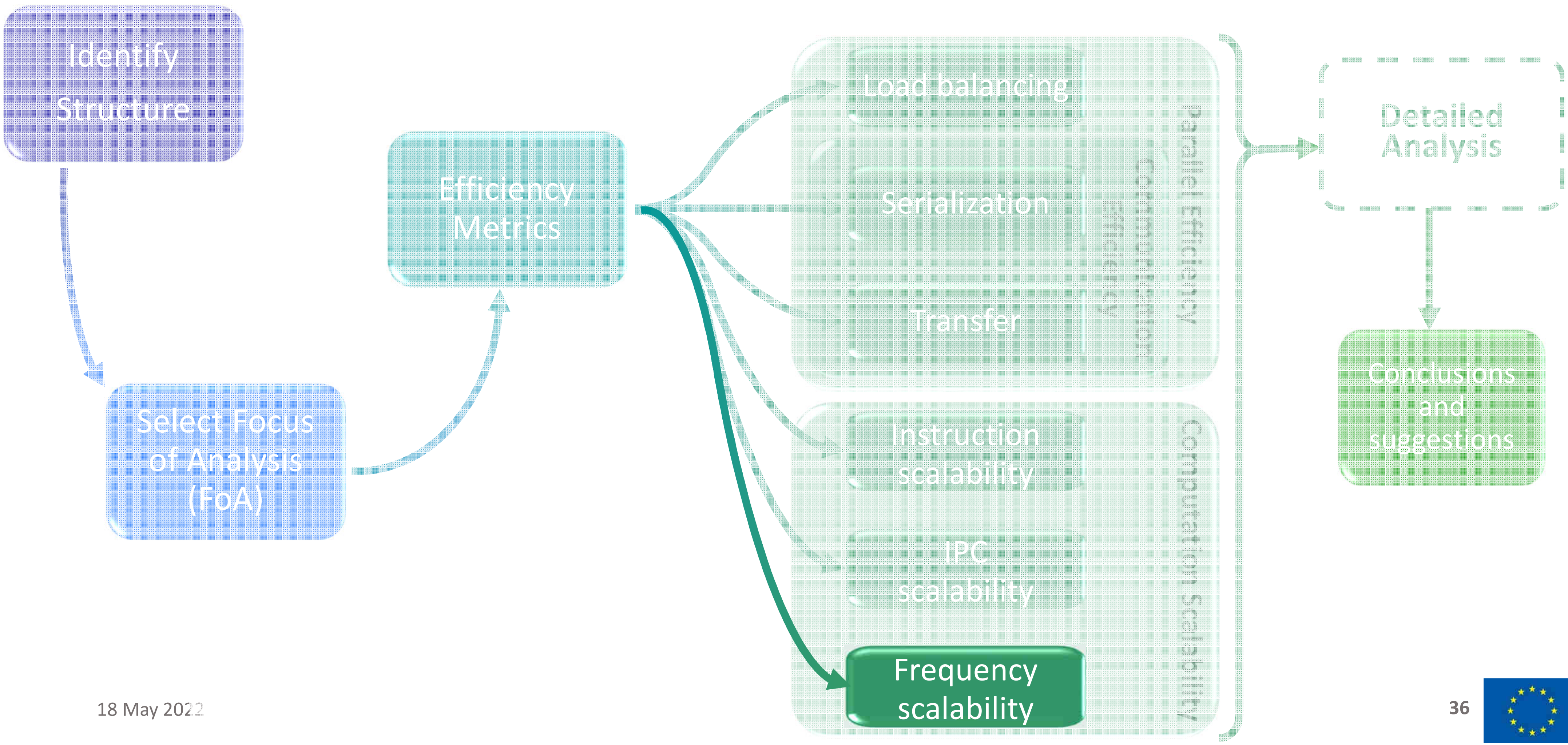
$$IPC\ Sc. = \frac{IPC_{current}}{IPC_{ref}}$$

- Relative metric, based on a reference case.
  - IPC_{ref} = Average IPC of the reference run
  - $IPC_{current}$ = Average IPC current run
- 100% otherwise

Interpretation: A value less than 100 indicates that the IPC for the specific core count is worse that of the reference case. May be caused by different locality behavior, contention on resources such as memory bandwidth, ...

A value above 100 indicates a higher IPC than the reference case this can be produced by cache effects for example.

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Frequency Scalability

Quantifies: How the frequency scales with respect to the reference case.

How it is computed: Ratio between average frequency on the current run and the average frequency on the reference run.

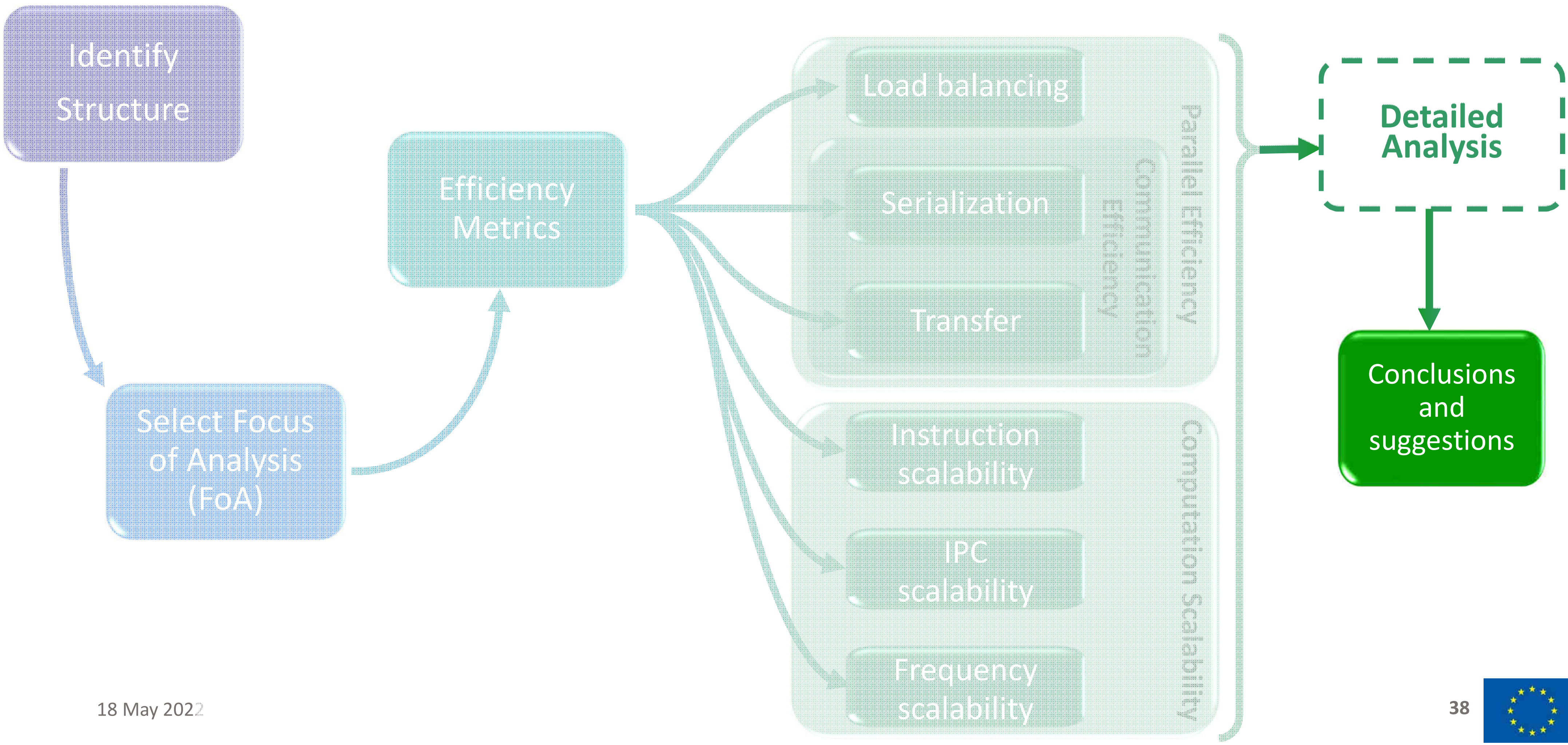
How it is computed:

$$Frequency\ Sc. = \frac{Freq_{current}}{Freq_{ref}}$$

- Relative metric, based on a reference case.
  - $Freq_{ref}$ = Average frequency on the reference run
  - $Freq_{current}$ = Average frequency on the current run
- 100% for the reference case

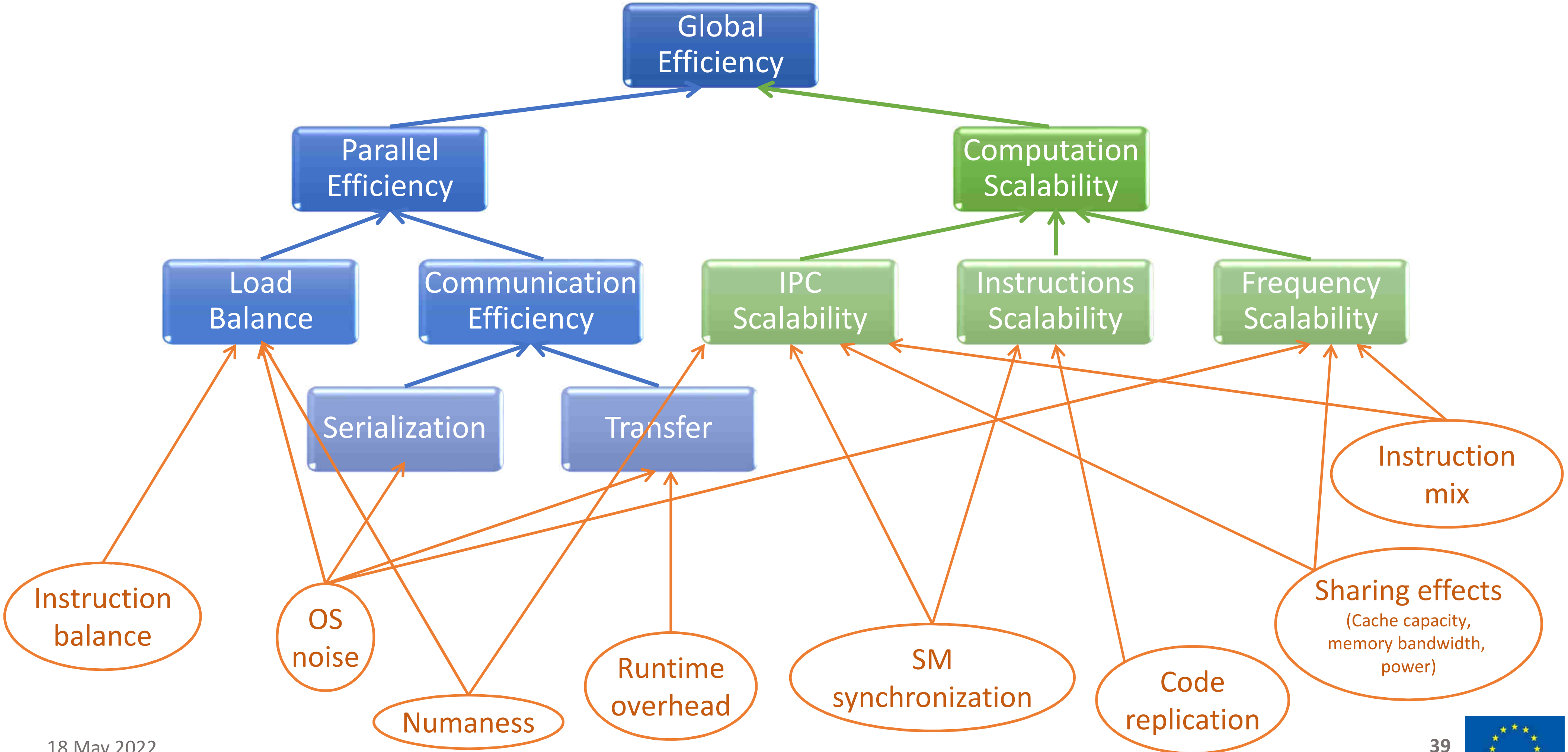
Interpretation: A value less than 100 indicates that the frequency is lower than the reference frequency. This may be caused by "preemptions", power management measures, ...

The journey



Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Factors and causes



Performance Optimisation and Productivity

A Centre of Excellence in HPC

Contact:

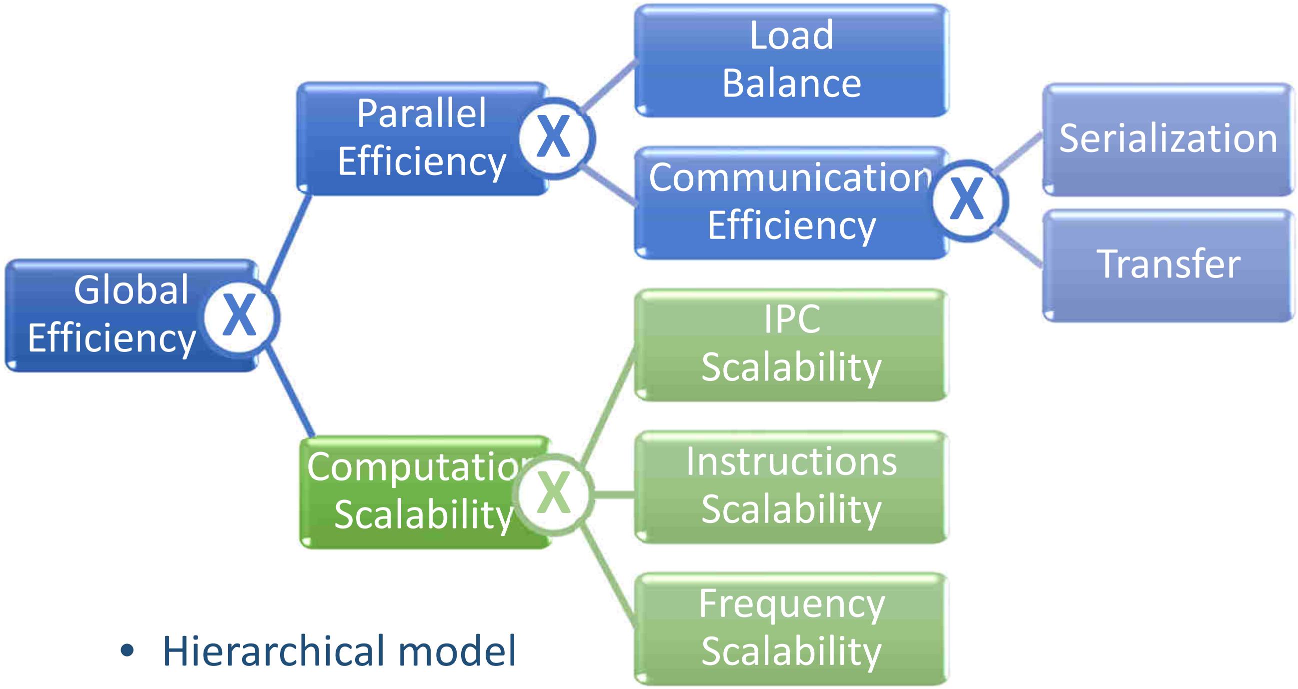
https://www.pop-coe.eu

mailto:pop@bsc.es

@POP_HPC

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676553 and 824080.

The Efficiency Metrics, summary



- Hierarchical model
- Multiplicative metrics
- Two kind of metrics:
  - Efficiency metrics
    - ➢ Absolute metrics
  - Scalability metrics
    - ➢ Relative to a base case
    - ➢ 100% for the base case

- **Parallel Efficiency**: The extent to which all resources in the system are kept active doing useful work
- **Load Balance**: The efficiency loss due to the global distribution of work among processes.
- **Serialization Eff.**: Inefficiencies caused by circular dependences or non-uniform imbalances
- **Transfer Eff.**: Efficiency loss related to the non instantaneous nature of communication mechanisms. Includes time to transmit the data over the physical channel and the overhead in the libraries.
- **Computation Scalability**: How the time spent computing scales with respect to the reference case.